IICA

CURSO CORTO SOBRE MANEJO DE DATOS DE INVESTIGACION USANDO SAS

CENTRO INTERAMERICANO DE DOCUMENTACION, INFORMACION Y COMUNICACION AGRICOLA

Documentación e Información Agrícola No.96

ISBN-92-9039-009-3
ISSN-0301-438X

INSTITUTO INTERAMERICANO DE COOPERACION PARA LA AGRICULTURA - IICA

CENTRO INTERAMERICANO DE DOCUMENTACION E INFORMACION AGRICOLA - CIDIA

CURSO CORTO SOBRE
MANEJO DE DATOS DE INVESTIGACION
USANDO SAS

PROYECTO DE INFORMACION AGROPECUARIA DEL ISTMO CENTROAMERICANO
-PIADIC-

San José, Costa Rica
1980

AYUDA FINANCIERA PARA LA TRADUCCION DE ESTA OBRA FUE SUMINISTRADA POR EL PROYECTO DE INFORMACION AGROPECUARIA DEL ISTMO CENTROAMERICANO (PIADIC) DEL INSTITUTO INTERAMERICANO DE COOPERACION PARA LA AGRICULTURA.

Esta publicación es traducción por Marigold Genis y Alvaro Garro de la versión original "A Short Course in Research Data Management Using SAS", que fue preparada por el IICA y publicada en julio de 1980 por Ronald W. Helms y James D. Hosking, de la Universidad de Carolina del Norte, U.S.A.

I N D I C E

L I S T A D E F I G U R A S

L I S T A D E C U A D R O S

PREFACIO: Antecedentes y Objetivos

Resumen

Existe un cuerpo de conocimientos técnicos y estrategias para el manejo de datos de investigación científica llamado Manejo de Datos de Investigación (MDI)*. El objetivo primordial de este curso es el de presentar una introducción a las principales técnicas y estrategias del MDI.

MDI es un arte muy aplicado; los métodos tienen sentido solamente si son utilizados en la práctica. Para usar las técnicas uno tiene que aplicar los programas y procedimientos de computación apropiados. En la actualidad el mejor juego de programas y procedimientos para este propósito es el Sistema de Análisis Estadístico (SAS). Aunque los participantes en este curso estén un tanto familiarizados con SAS, vamos a presumir que se tienen pocos antecedentes sobre su uso en el MDI moderno. Por consiguiente, la primera parte de este curso es un introducción a la programación del SAS, con especial énfasis en las facilidades que ofrece, particularmente en el MDI.

En resúmen, los objetivos son: (1) la presentación de técnicas de programación del SAS útiles en el manejo de datos de investigación; y (2) la presentación de las principales técnicas y estrategias para el manejo de datos de investigación.

El advenimiento de la computadora condujo muy rápidamente a la creación de nuevas disciplinas, incluyendo la ciencia de la computación, el análisis de sistemas y varias combinaciones y subjuegos de estas disciplinas, relacionados específicamente con el manejo y procesamiento eficiente y confiable de datos. Tanto la comunidad científica como la comercial desde un principio aprovecharon la tecnología de computación en la década de los cincuenta. Los usuarios comerciales, reconociendo que el procesamiento de datos representa en sí mismo una gran parte del costo de las operaciones de computación,

---

* Research Data Managemente (RDM)

apoyaron el estudio de sistemas y técnicas eficientes para el procesamiento de sus propios datos. Los productores de programas y procedimientos altamente eficientes para el procesamiento de datos comericales, fueron prontamente recompensados; puede citarse como ejemplo el evidente éxito de las compañías que producen paquetes eficientes de clasificación.

La comunidad científica, en general, tomó una dirección diferente. En las décadas de los cincuenta y los sesenta los problemas de computación científica más importantes consistían en una flata de programas y procedimientos computacionales. Elevados recursos se asignaron al desarrollo de programas y procedimientos científicos generales, incluyendo, en el área estadística, los grandes paquetes tales como BMD, SPSS, BMPD y SAS72. De principios a mediados de la década de los setenta los profesionales en el campo de la "computación estadística" comenzaron a darse cuenta de que la capacidad de los científicos para colectar datos superaba la habilidad del personal de computación para procesar y manejar esos datos eficazmente. Una vez que los datos fueran procesados en un formato conveniente, los paquetes estadísticos podrían producir con facilidad la mayoría de las estadísticas requeridas.

De esta situación nació una nueva subdisciplina: Manejo de Datos de Investigación (abrevidado MDI), que consiste en el estudio de los problemas peculiares del manejo y procesamiento de datos de investigación científica, junto con las estrategias y métodos desarrollados para la solución de estos problemas.

Algunas de las técnicas del MDI son francas adaptaciones de manejo de datos y métodos de análisis de sistemas comerciales. La clasificación de datos científicos es muy parecida a la de datos comerciales, por ejemplo, y los mismos juegos de programas y procedimientos se usan para ambos. Sin embargo, hay importantes diferencias entre los datos de investigación y los datos comerciales, las cuales requieren distintas estrategias de manejo.

Una de las diferencias importantes es el efecto de los errores de los datos. Por ejemplo, compárese un sistema bancario para el procesamiento de transacciones de cuentas de cheques, con un "sistema" para el procesamiento de datos de un gran estudio de ciencias sociales. Un tipo de error al azar

de 1% en el sistema de cuentas de cheques sería intolerable; el mismo tipo de error en el estudio de ciencias sociales podría ser literalmente imposible de detectar. Así, las estrategias para el control de calidad son diferentes en el manejo de datos comerciales y de investigación.

Una segunda diferencia importante involucra los tamaños y formas de los juegos de datos. Los juegos de datos comerciales (como el caso de las cuentas de cheques, por ejemplo) a menudo tienen relativamente pocas "variables" o "campos" (típicamente 6 o 10 campos para datos de cheques) pero un volumen muy grande de registros para ser procesados. En contraste, los datos científicos frecuentemente tienen decenas, o cientos de variables y menos registros ("casos") que un juego de datos comercial comparable.

El Manejo de Datos de Investigación logró varios adelantos importantes en la década de los años setenta. Hubo gran desarrollo en los paquetes de programas y procedimientos, particularmente las facilidades de manejo de datos del SAS72, SAS76 y SAS79. Hubo también investigadores y profesores que empezaron a diseñar nuevas técnicas (por ejemplo los sistemas basados en tablas, subsistemas de inventario y "statusbytes", todos en las Clínicas de Investigación Lipid "Sistema de Manejo de Datos Visita 2", Helms[2] y a organizar y consolidar los conocimientos en el área. Ejemplos de esto último incluyen a Bob Teitel en el Instituto Urbano, quien presentó una serie de ensayos en el Interface Symposia, y al presente autor y Wendell Smith en la Universidad de Carolina del Norte, quienes desarrollaron un sub-curriculum de Postgrado en el Manejo de Datos de Investigación.

Como resultado de los esfuerzos de estos y otros investigadores, en 1980 el MDI comenzó a ser reconocido como una disciplina genuina, generalmente dentro del área de computación estadística. Existe un acervo de conocimientos, una colección de estrategias y técnicas, para solucionar los importantes problemas que encara el manejo de datos de investigación.

El objetivo primario de este corto curso es presentar una introducción a las técnicas y estrategias más importantes del MDI. MDI es en sí mismo una ciencia/arte muy práctica y aplicada; sin embargo, un conocimiento efectivo requiere que el estudiante practique las técnicas y estrategias con los

propios paquetes de programas y procedimientos, y sobre datos reales. En este momento el mejor paquete de programas y procedimientos disponible para el manejo de archivos pequeños o medianos de datos se halla en el SAS79[6].

Por consiguiente, deseamos presentar, como parte del texto, un cursillo sobre programación de SAS, con énfasis en las instrucciones y procedimientos importantes para el MDI.

CAPITULO 1

INTRODUCCION

1. Un Compendio de Manejo de Datos de Investigación: ¿Qué es MDI?

El MDI es un conjunto de métodos y estrategias para el manejo de datos de investigación científica. Esta definición es simple pero para entender qué es en realidad el MDI es indispensable conocer algo de manejo de datos con computadoras y algunos de los factores que distinguen los datos de investigación de, digamos, los datos comerciales. Estos temas constituyen la sustancia de la última parte de este texto.

Las actividades del MDI pueden ser clasificadas dentro de cinco categorías generales:

1. Planificación
2. Diseño e Implementación del Sistema MDI
3. Procesamiento de datos de "Producción"
4. Proceso de datos del Archivo de Análisis
5. Computación Estadística

Estas categorías se mencionan aproximadamente en el orden en que las actividades se ejecutan para un proyecto en particular.

La planificación es obviamente una función gerencial e incluye la estimación de los recursos que se necesitan (fondos, personal, tiempo calendario), adquisición de recursos, planificación de actividades, otros (Helms[3]). El diseño e implementación involucran Análisis del Sistema MDI para la fase de diseño y los esfuerzos de programadores entrenados en MDI para la implementación (programación, eliminación de errores, comprobación, documentación).

Una vez que un sistema de MDI se implementa, es típicamente usado u operado por procesadores de datos, individuos con la personalidad y habilidad para manejar los datos cuidadosamente, para alimentar los datos al Sistema MDI computarizado y para usar el sistema en la detección y corrección de errores, y producir registros maestros "limpios" y bien organizados. Esto es la "generación" en el procesamiento de datos, que tiende a ser trabajo rutinario y tedioso.

Los archivos maestros resultantes de la producción de procesamiento de datos son por lo general inconvenientes para análisis estadístico, en parte porque un sistema MDI es usualmente diseñado para "optimizar" la fase de producción en el procesamiento de datos. Además, los archivos maestros pueden tener una estructura muy complicada para uso directo por programas de análisis estadístico. Antes de que la computación estadística pueda llevarse a cabo (fase 5), usualmente se tienen que ejecutar varias tareas para producir archivos de análisis de archivos maestros: esta es la fase de "Proceso de datos del Archivo de Análisis".

Finalmente, se tienen archivos que pueden ser introducidos directamente a un paquete estadístico tal como SAS, para ejecutar las computaciones estadísticas deseadas, última fase del proceso del MDI.

Esta discusión, es, desde luego, muy general. En un proyecto específico, una o más de estas fases pueden ser tan "fáciles" que están esencialmente incorporadas dentro de otra fase. En otros proyectos, más grandes y complejos, cada fase puede tener un costo significante y requerir un intervalo de calendario apreciable.

Temas a ser discutidos

Hay, claramente, mucho más material en el MDI que el que puede ser cubierto en un texto de esta magnitud. Los temas que se han de cubrir aquí son:

1. Técnicas de programación del SAS
2. Componentes de un Sistema MDI General
3. Diseño de un Sistema MDI
4. Planificación para el MDI

Este curso corto en realidad consiste en un repaso bastante comprensivo de las facilidades de programación del sistema SAS y se apoya fuertemente en textos sobre el SAS79. (SAS79 es nuestra abreviatura de la "Guía del Usuario del SAS[6], edición de 1979).

En la segunda parte se presenta un diagrama del flujo general del Sistema MDI y se discute brevemente cada uno de los subsistemas componentes.

El tercer tema principal, Diseño del Sistema MDI, se omite en este texto, debido en parte a lo corto del curso y en parte al hecho de que el diseño del MDI es todavía un arte medianamente primitivo.

La planificación para el MDI se discute brevemente en el Apéndice 1. Esta es muy similar a la planificación para otros tipos de programas y procedimientos y el material en esta sección depende mucho del trabajo de otros gerentes de programas y procedimientos. (Brooks[1], Metzger[5]).

CAPITULO 2

PROGRAMACION SAS PARA RDM

2.1 Breve estudio general del procesamiento SAS

A pesar del nombre "Sistema de Análisis Estadístico", SAS es en realidad un lenguaje de procesamiento de datos con facilidades de análisis estadísticos extensos. SAS es diferente a la mayoría de los otros lenguajes de programación, tales como FORTRAM y PL/I.

Una diferencia es que hay básicamente dos tipos de "programas" SAS. Uno se llama "DATA Step" porque comienza con la instrucción de DATA y está específicamente orientado a crear un nuevo "Archivo" (un archivo en un formato especial).

El otro tipo se llama "PROC Step" porque comienza con la instrucción PROC de SAS (Procedimiento); la mayoría de los pasos PROC involucran subprogramas SAS para realizar algún tipo de computaciones estadísticas.

Puesto que este es un texto sobre el manejo de datos, el principal interés estadístico radica en los DATA Steps.

Un DATA Step contiene un programa que permite la entrada de uno o más archivos y la salida (usualmente) es un archivo SAS. Las instrucciones de los programas superficialmente se parecen mucho a instrucciones PL/I. Los mismos símbolos son usados, en su mayor parte, para operadores numéricos tales como multiplicación y división, y cada instrucción termina con un punto y coma (";"). Pero el programa para un SAS DATA Step es estructuralmente diferente. El modo básico de operación consiste en que SAS alimenta un registro de datos (una observación de un archivo SAS o uno o más registros de datos de un archivo externo), ejecuta las instrucciones del programa una vez, y entonces automáticamente escribe la "observación" SAS al archivo SAS producido.

El programador SAS no está obligado a escribir instrucciones para leer y escribir registros de datos -todo esto es realizado semiautomáticamente por SAS. El programador puede, desde luego, tomar control de estas funciones hasta donde así lo desee o se requiera.

La entrada y salida (I/O) de información es uno de los aspectos más problemáticos de la programación; sin embargo, el SAS se ocupa de todo el I/O del y para los archivos SAS. Por tanto, el SAS puede ser un lenguaje extremadamente fácil y eficiente para procesar datos una vez que esos datos han sido inicialmente almacenados en los archivos SAS.

Aún más, SAS ofrece grandes facilidades para incorporar documentación de los datos directamente en los archivos SAS. Estas y otras características hacen del SAS aproximadamente un grado de magnitud mejor que los lenguajes de programación para el manejo de datos de investigación y otros datos.

2.2 Archivos OS, Bases de Datos SAS y Archivos SAS

El propósito básico de esta sección es eliminar, tan pronto sea posible, la confusión que suscitan los múlitples usos de la palabra Archivos. En términos generales, un archivo es cualquier grupo de datos; sin embargo, aquí no ocuparemos con tipos de archivos muy específicos.

Un archivo OS es un archivo que se accesa por la vía del Sistema Operativo.

Como ejemplo se incluyen:

1. Un grupo de tarjetas en un archivo "SYSIN"
   ```
   //SYSIN DD *
   Grupo de tarjetas = Archivo OS
   ```
2. Una colección de registros en una cinta magnética
   ```
   //TAPESET DD DISP=OLD,UNIT=TAPE,DSN=ABC.DEF
   ```
3. Una colección de registros en un disco magnético
   ```
   //DISKEX DD DISP=OLD,UNIT=DISK,DSN=QRS.TUV
   ```
4. Un archivo "SYSOUT" es en realidad un archivo de disco. Después de que se ha terminado la ejecución, el Sistema Operativo copia cada archivo SYSOUT asl impresor y luego suprime el archivo del disco (ejemplo: `//SYSPRINT DD SYSOUT=A,DCB=(RECFM=VBA,BLKSIZE=200)`

Un archivo puede contener datos en cualquiera de varios formatos; a nosotros nos interesa principalmente en dos formatos:

"EBCDIC" (Código de Intercambio Decimal de Codificación Binaria Extendida). Las tarjetas son "perforadas", básicamente, en este formato; los Archivos SYSOUT están en el formato EBCDIC.

El formato "Interno" o "punto flotante hexadecimal" es usado por SAS para sus archivos. En SAS los números tiene que ser convertidos de EBCDIC (u otro código) al "Código de punto flotante hexadecimal interno" o "Formato interno" antes de procesarlos como números. El proceso de conversión es, a menudo, relativamente caro.

Una base de datos SAS es un archivo OS (en disco, algunas veces en cinta) escrito por SAS en un formato especial. Una base de datos SAS contiene:

* Uno o más archivos SAS (discutido más abajo)
* Información acerca de cada archivo SAS en la base de datos (Esto se discute más adelante).

Un archivo SAS consiste en una matriz de datos, más información especial sobre cada variable.

Una matriz de datos estándar (llamada también un "archivo plano") se presenta en el siguiente formato Fig.1):

Presentación de una matriz de datos estándar

Columnas = Variables

Datos del:	Variable 1	Variable 2	...	Ultima Variable
1er. sujeto			...	
2do. sujeto			...	
3er. sujeto			...	
...	...	...	...	...
no. sujeto			...	

Fig. 1 Matriz de datos

Una línea de la matriz de datos contiene todos los datos de un sujeto, esta línea se llama una "observación", un "caso", un "registro", etc.

El archivo SAS contiene una matriz de datos (más información tal como nombres de las variables, formatos, otros).

El SAS crea (escribe) un archivo SAS una observación entera cada vez (una línea cada vez). SAS lee un archivo SAS, una observación cada vez (todas las variables de una sola vez).

Dentro de un archivo SAS, toda observación tiene la misma estructura. Las variables están en el mismo orden. Una variable tiene la misma longitud (número de "bytes" para fines de almacenamiento) a través de todas las observaciones. Solo los valores de los datos difieren de una observación a la siguiente.

El SAS puede poner varios archivos SAS en una Base de Datos SAS. Los archivos pueden ser completamente diferentes, es decir, sin ninguna relación el uno al otro. Desde luego, también pueden estar muy relacionados entre sí.

Para referirse a un archivo SAS dentro de una base de datos SAS, tiene que indicarle al SAS dos cosas:

* La información para localizar la base de datos (i.e., el archivo OS que contiene la base de datos). Esto es un nombre en una instrucción DD. Por ejemplo, en la siguiente instrucción OS
  //PERM DD DISP= OLD, UNIT= DISK, DSN= ABC.DEF
  "PERM" es el nombre de la base de datos SAS (dd name) y "ABC.DEF" es el OS DSM (le dice al OS en donde encontrar el archivo).
* El nombre del archivo SAS en la base de datos: e.g., en la instrucción SAS DATA PERM.MYDATA
  "MYDATA" es el nombre del archivo SAS dentro de la base de datos PERM. (Aquín se ignora el uso de especificaciones del archivo de la forma DATAMYDATA, i.e., aquellos sin un nombre de base de datos, porque su uso es una mala práctica de programación).

Existe una base de datos SAS temporal, especial, llamada WORK. WORK es suprimida, junto con cualesquiera otros archivos SAS almacenados en ella, al final del trabajo. Work es útil para almacenar resultados preliminares. El nombre de un archivo en WORK es igual a los nombres de los archivos en otras bases de datos, e.g., WORK.MYDATA.

SAS es una colección de un gran número de programas de computadora y un programa "supervisor". El supervisor llama a los varios programas a medida que se necesitan. Un trabajo típico SAS contiene varios "Pasos SAS". Las instrucciones DATA o PROC son simpre la primera instrucción de un Paso SAS.

Un "Paso" SAS comienza con una instrucción PROC o DATA y termina en la última instrucción anterior de la próxima instrucción PROC o DATA. Esta estructura se muestra a continuación Fig.2.

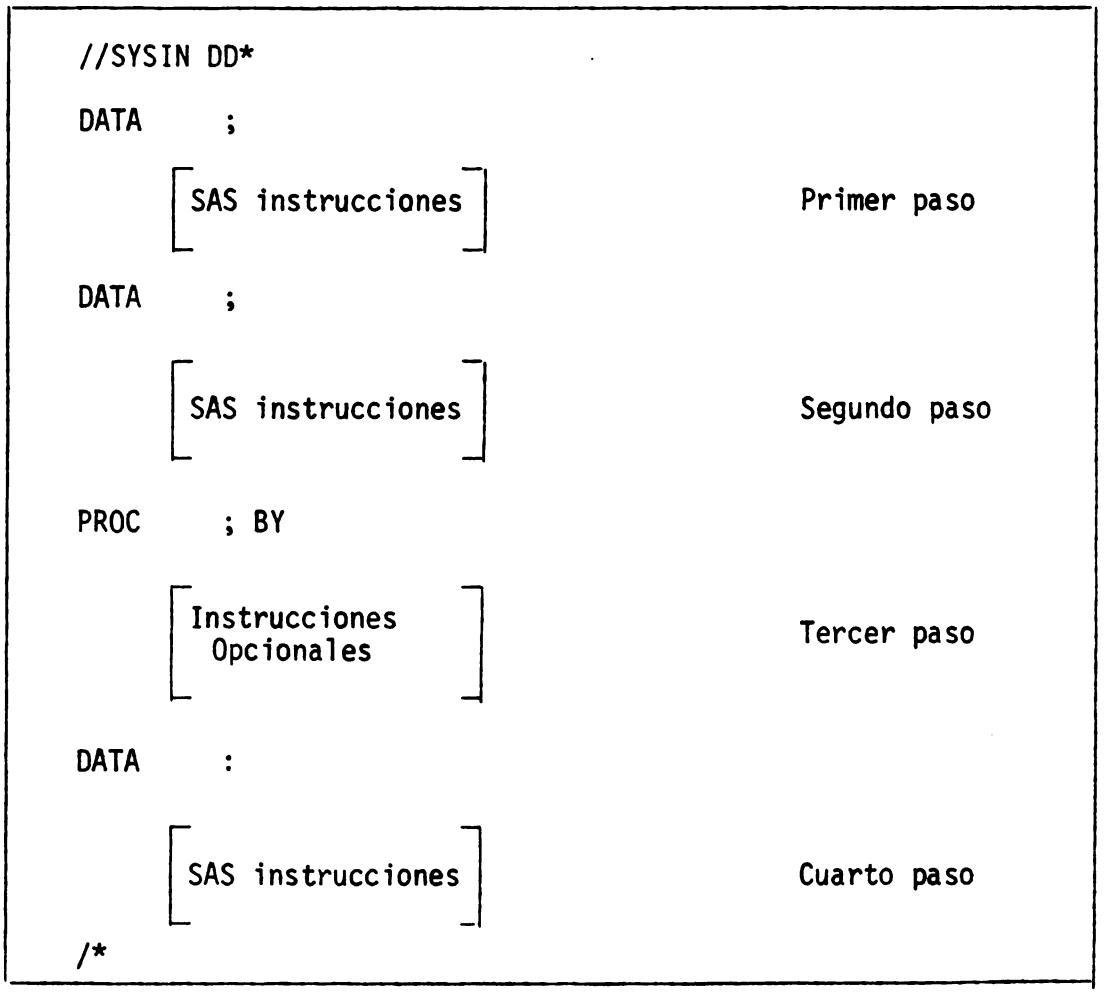


Figura 2 Estructura de "Pasos SAS"

En el procesamiento de un trabajo SAS, el supervisor del SAS:

* Identifica el primer paso;
* Llama al compilador para traducir las instrucciones SAS a instrucciones ejecutables por la computadora (en su mayoría, llamadas subrutinas);
* Traspasa control al programa recopilado;
* Si aparece un error en la ejecución del programa recopilado el supervisor toma control y decide: a) terminar el procesamiento, o (b) continuar procesando con OBS = 0 (que luego se explica) o (c) continuar con el procesamiento normal;
* (Procesamiento normal): Imprime "NOTES" acerca de la ejecución del paso;
* (Procesamiento normal): Identifica el próximo paso y vuelve al paso de compilación.

Un "paso DATA" (paso que comienza con una instrucción de DATA) típicamente involucra la creación de un nuevo archivo SAS (dentro de una base de datos SAS específica). Los datos de entrada pueden provenir de un archivo OS (EBCDIC) externo o de uno o más archivos SAS con bases en datos SAS específicos.

Un "paso PROC" (paso comenzando con una instrucción PROC) usualmente involucra el análisis de un archivo SAS (que fue creado con anterioridad). El resultado es generalmente una información impresa, en vez de nuevos archivos SAS.

Habrá muchas excepciones: algunos pasos DATA son "programas que generan reportes", los cuales producen un registro impreso como principal información producida. Algunos PROCS, tales como PROC SORT, principalmente procesan datos.

2.2.1 Ejecución SAS de Pasos de Datos

Después de recopilar un programa de un paso DATA el supervisor SAS ejecuta el programa una vez por cada observación en los datos. Después de ejecutar todas las instrucciones del programa, SAS escribe la observación, en "formato interno", al archivo SAS producido. Un Programa típico de la fase de entrada de datos puede diagramarse de la siguiente manera (Fig.3).

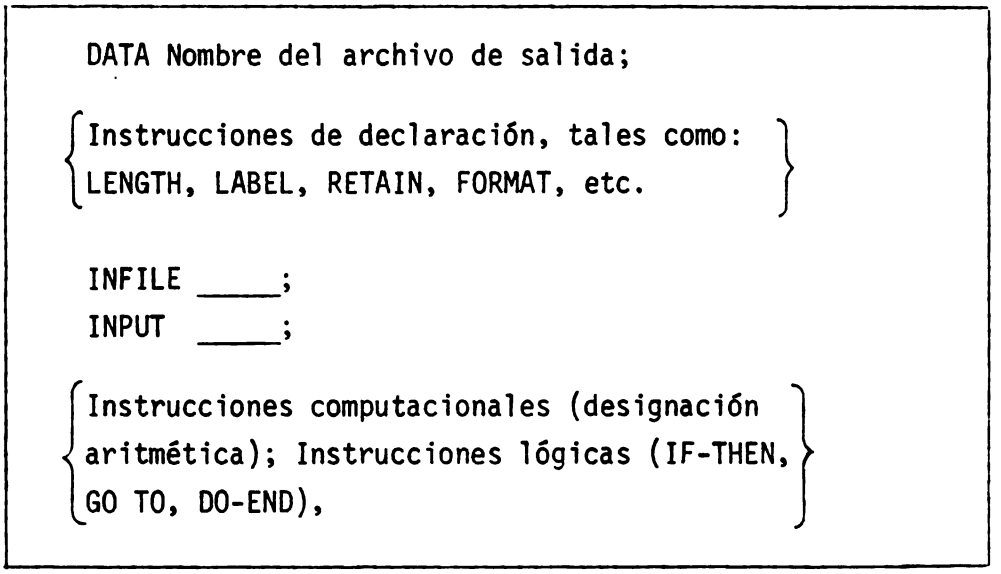


Figura 3. Diagrama de programa típico de entradas de datos

Típicamente, SAS repite la siguiente secuencia para cada observación de entrada:

a. Ejecuta la instrucción INPUT (alimenta las variables indicadas en la instrucción de acuerdo a las especificaciones contenidas en ella).
b. Ejecuta las instrucciones computacionales y lógicas.
c. Al "final" del programa, SAS escribe la observación (los valores de todas las variables para esta observación) en el archivo SAS indicado en la instrucción DATA.

Se repite toda la secuencia para cada observación. Como luego se observará, el programador puede cambiar esta secuencia en muchas formas. Sin embargo, a menos que sea cambiado por el programador, SAS usa la secuencia arriaba descrita.

2.3 La Fase de Entrada de Datos (DIP)

La primera fase computarizada del procesamiento de datos es la Fase de Entrada de Datos, la cual tiene los siguienes objetivos:

a. Alimentar datos de un archivo OS (EBCDIC, tarjeta o cinta) "externo" y crear un archivo SAS dentro de una base de datos SAS específica.

b. Realizar pruebas preliminares en busca de errores graves tales como, tarjetas extraviadas, valores omitidos, otros.
c. Crear documentación interna de los archivos, tales como los nombres de las variables, calificativo extendido de las variables, formatos de variables, otros.

El diagrama de flujo básico de un programa de la fase de entrada de datos (Figura 4) es muy simple. Cualesquiera manipulaciones complicadas de datos son pospuestas hasta después de que los datos se almacenen en un archivo SAS. Debido a la gran cantidad de documentación, los programas SAS para la fase de entrada de datos son a menudo largos, pero al grano.

2.3.1 Un ejemplo de la Fase de Entrada de Datos SAS

La Figura 5 presenta un ejemplo de un programa de la Fase de Entrada de Datos SAS. El formato de los datos alimentados y una lista de unas cuantas tarjetas se encuentran en el Cuadro No.1. En este ejemplo el formato de datos es muy simple: hay una tarjeta por cada observación y el formato de las tarjetas es fijo.

Este ejemplo es bastante típico de la mayoría de los programas DIP SAS con la excepción del hecho de que la mayoría de los programas tienen una entrada de datos más complicada. Los siguientes párrafos discuten la programación de la Fase de Entrada de Datos y sus instrucciones SAS, en términos del ejemplo dado. La extensión de estas ideas a un caso más general es directa.

Instrucción JOB: Los detalles del JCL JOB de la instrucción no se incluyen. La instrucción JOB varía marcadamente de instalación a instalación. El lector debe obtener los detalles del JCL que se emplea, de los funcionarios de la instalación de cómputo.

Listado (ver página 19).

Instrucción PERM DD: Esta instrucción identifica la base de datos SAS que contendrá el archivo SAS creado por el programa. Al igual que la instrucción JOB, los detalles de la instrucción DD varían mucho de una instalación a otra; nuevamente se le recomienda al lector dirigirse a

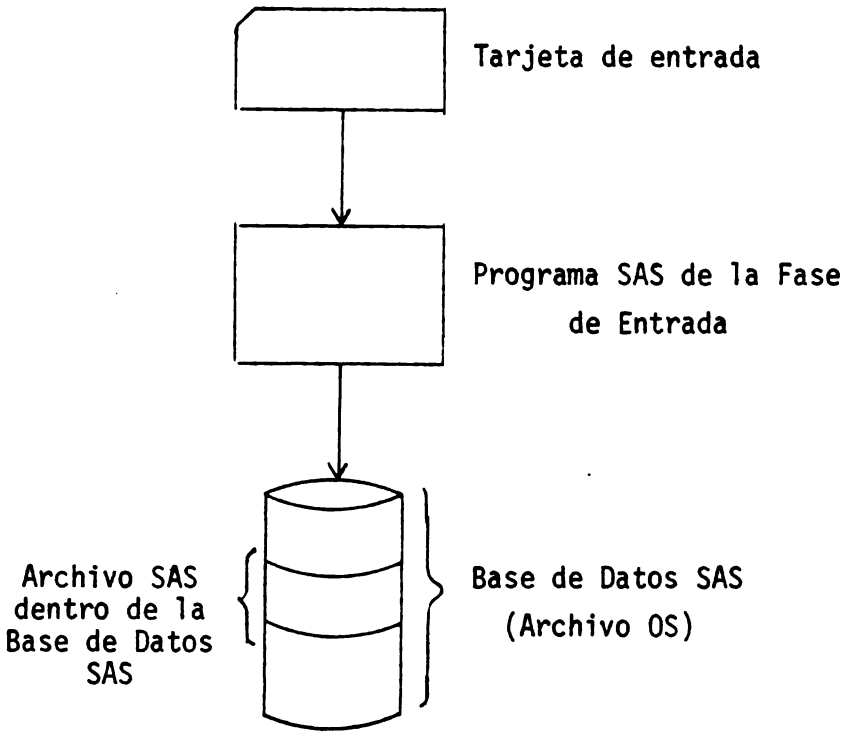


Fig. 4. Un diagrama de flujo simple de una Fase de Entrada de Datos

los funcionarios del Centro de Cómputo local para determinar el mecanismo para la creación de archivos permanentes en disco y el JCL local para determinar el modo de usarlos.

Instrucción TARJETAS DD: El Archivo SYSIN, cuyo nombre es TARJETA contiene los datos de entrada para el programa.

Los datos también pueden colocarse con el programa SAS, precedidos por una instrucción SAS "CARDS"; pareciera ser una buena práctica de programación el separar el programa de los datos, como aquí se ha hecho.

Frecuentemente se prueba un programa con un pequeño grupo de tarjetas de datos y una vez que se ha hecho la prueba se alimenta un archivo más grande contenido en disco o cinta con imagen de tarjetas.

```
//      JOB
//      EXEC SAS
//PERM DD DISP=OLD, UNIT=DISK,DSN=ABC.DEF.GHI
//*PERM WAS CREATED IN AN EARLIER JOB
//TARJETAS DD *
              Tarjetas de datos
//SYSIN DD *
TITLE EXAMPLE OF SAS DATA INPUT PHASE;
TITLE2 STORE PHYSICAL FITNESS DATA IN PERM.PFD79
DATA PERM.PFD79(LABEL=1979 PHYSICAL FITNESS DATA);

       INFILE TARJETAS;
       LENGHT        NAME$20
                     SEX $1
                     HEIGHT
                     WEIGHT 4;
       LABEL         HEIGHT=HEIGHT OF SUBJECT IN CM
                     WEIGHT=WEIGHT OF SUBJECT IN KG
                     AGE=AGE OF SUBJECT IN YEARS
                     ;
       FORMAT        DATEBORN
                     SURVDATE YYMMDD8.;
       INPUT         CARDTYPE $1-5
                     IDNUM     6-9
                     SEX      $32
                     HEIGHT    4-38
                     WEIGHT    40-44
                  @ 46 DATEBORN  YYMMDD6.
                  @ 53 SURVDATE  YYMMDD6.;
       AGE=          (SURVDATE-DATEBORN)/365,25;
PROC PRINT;
       TITLE3 LISTING OF INPUT DATA;
PROC SORT; BY NAME;
PROC PRINT; ID NAME;
       TITLE LISTING OF INPUT DATA SORTED BY NAME;
```

Fig. 5. Ejemplo de un Programa SAS de la Fase de Entrada de Datos (El formato de los datos se describe en el Cuadro No. 1).

Al separar el programa de los datos, el cambio de fuente de alimentación de tarjetas o cinta o disco solamente implicaría cambios a la instrucción DD; no se requerirán cambios SAS. También hay otras razones para mantener los datos y los programas en archivos separados.

FORMATO

Columnas	Descripción	
1-3	PFD	Descriptor de tipo de tarjeta (nemónico)
4-5	79	Año en que se recogieron los datos
6-9	IDNUM	Número de identificación del tema o persona
11-30	NAME	Nombre del sujeto o tema (apellido, nombre, inicial)
32	SEX	Codificado como M o F
34-38	HEIGHT	Altura, en cm.
40-44	WEIGHT	Peso en Kg (xxx.x)
46-51	DATEBORN	Fecha de nacimiento (yymmdd)
53-58	SURVDATE	Fecha de la encuesta (yymmdd)

Cuadro No.1. Formato de Datos de Buena Salud Física (PFD 79) y Lista de Datos

Instrucción SYSIN DD: Esta instrucción de JCL define (precede) el archivo que contiene el programa SAS.

Los archivos LOG y PRINT: Los programas SAS típicamente producen dos archivos de salida impresos nombrados archivos LOG y PRINT, respectivamente. Los listados del programa, NOTEs del supervisor SAS, y los mensajes de error SAS se imprimen en el LOG. Seguidamente del LOG, el archivo PRINT contiene información impresa producida por cualquier PROC(s) que se ejecuten. El programa de entrada de datos puede colocar información producida en cualquiera de los archivos.

Las instrúcciones TITLE y TITLE2 en el ejemplo, especifican los títulos a imprimirse por SAS en la parte superior de cada página del archivo PRINT (no en el archivo LOG). Las instrucciones se describen el SAS79[6] en la p. 18. El lector debe estudiar esta descripción.

1.10	11.20	21.30	31.40	41.50	51.60
1 2 3 4 5 6 7 8 9 10	11 12 13 14 15 16 17 18 19 20	21 22 23 24 25 26 27 28 29 30	31 32 33 34 35 36 37 38 39 40	41 42 43 44 45 46 47 48 49 50	51 52 53 54 55 56 57 58
PFD79 347	CHRISTIANSEN, DAVE H	M 180	88.5	4804	4 790126
PFD79 591	HOSKING, JAMES D.	M 178	79.3	5103	1 790214
PFD79 568	HELMS, RONALD W.	M 180	93.0	4110	30 790127
PFD79 932	PIGGY MISS F.	F 122	45.4	5902	31 790607
PFD79 427	FROG, KERMIT G.	M 61	0.45	7604	31 790823
PFD79 023	gonzo, (NFN) (NMN)	64		6506	25 790823

Cuadro No. 2. Listado del ejemplo

Además de rotular la información impresa producida, las instrucciones TITLE forman una parte importante de la documentación de las actividades del procesamiento de datos.
Las instrucciones COMMENT, dentro del programa, también sirven para estos fines de documentación.

2.3.2 Instrucción DATA

La instrucción DATA en el ejemplo, la cual es una instrucción típica de la fase de entrada de datos, ejecuta las siguientes funciones:

a. Es la primera instrucción de un paso DATA.
b. Identifica la base de datos SAS producida ("PER") y el archivo SAS producido ("PFD79").
c. Provee cierta otra información acerca de los archivos de salida, tal es el caso del rótulo (label) de un archivo, el cual contiene una descripción del archivo más larga de la que puede ser contenida por un nombre de 8 caracteres.

En su forma más general, las instrucciones DATA pueden ser bastante complicadas. La forma más común, ilustrada en el ejemplo, debe ser imitada por el lector hasta que se hayan discutido formas más avanzadas.

2.3.3 La instrucción INFILE

La instrucción INFILE le especifica al supervisor SAS el nombre del archivo OS que contiene los datos de entrada.

(Nota: el incluir los datos en el programa después de la instrucción "CARDS", lo cual no se recomienda, es una alternativa al uso del INFILE).

Las instrucciones INFILE pueden ser bastante complicadas en algunos casos; el SAS79[6], un manual de referencia, describe todas las opciones posibles.

El leer ahora acerca del INFILE en SAS79[6], no se recomienda; es preferible esperar hasta que se hayan cubierto temas más avanzados.

Por ahora simplemente imítese el uso mostrado en el ejemplo, pero siéntase libre de crear sus propios nombres dd (para reemplazar "TARJETAS").

2.3.4 Puntos sobre Estilo de Programación

Las reglas para la creación de nombres SAS, tales como nombres de variables, nombres de archivos, nombres de bases de datos, nombres dd, otros, se describen en las páginas 7 y 8 del SAS79. El lector debe repasar esta descripción ahora.

No se debe usar nombres que comienzan o terminen con subrayados, tales como _ALL_, _N_, etc. SAS tiene cierto número de nombres especiales con este formato. Así, si aplicara este formato podría accidentalmente usar una palabra "mágica" y crear algunos problemas inesperados.

Sangrías. Nótese que todas las instrucciones en el ejemplo que sigue a la instrucción DATA están con sangría. Este tipo de formato de programa no es requerido por SAS, pero es altamente recomendado, como lo son los otros tipos de sangrías en el ejemplo. En este caso, la sangría nuestra de un solo vistazo cuales informes están incluídos en cada paso SAS del programa. Tal sangría, por consiguiente, parcialmente documenta la estructura del programa. Un apropiado formato de programa es también una herramienta muy útil en la eliminación de errores.

2.3.5 La instrucción LENGTH

La instrucción LENGTH es una instrucción de declaración muy importante, que especifica al compilador SAS.

a. El tipo (carácter o numérico) de cada variable especificada en la instrucción; y
b. La longitud de cada variable en la lista, es decir, el número de bytes a usarse para almacenar un valor de la variable.

Una variable es declarada carácter si el símbolo "$" sigue su nombre en la instrucción LENGTH; cualquier nombre de variable en un LENGTH no seguido por un "$" es número.

Cada variable tiene asociada una longitud, en bytes. Las variables numéricas deben tener por lo menos dos bytes; su longitud máxima es de 8 bytes (aproximadamente 15 dígitos decimales más de un exponente 1-byte. Las variables de caracteres pueden ser de cualquier longitud, desde 1 byte hasta 200 bytes.

El compilador SAS determina tanto el tipo como la longitud de una variable a la primera aparición del nombre de la variable en el programa. Si en este punto no hay información explícita acerca del tipo y/o longitud, SAS usará numérico como el tipo implícito y/o 8 bytes como la longitud faltante.

En el ejemplo, se declaran solamente cuatro variables explícitamente, en cuanto a tipo y longitud en la instrucción LENGTH.

SAS hará de DATEBORN y SURVDATE variables numéricas cuando los encuentre en la instrucción FORMAT, porque se usa un formato numérico (YYMMDD8.). Las longitudes serán las implícitas, 8 bytes.

SAS hará del CARDTYPE una variable carácter cuando la encuentre en la instrucción INPUT debido al $ que sigue al nombre.

SAS determinará que el largo del CARDTYPE es de 5 bytes porque ese es el número de bytes en el campo de entrada.

IDNUM será declarado numérico al encontrar su nombre en la instrucción INPUT, porque no hay "$" o formato de carácter siguiendo el nombre. Sin importar la longitud del campo de entrada, las variables numéricas tienen una longitud implícita de 8 bytes. Así, IDNUM tendrá una longitud de 8 bytes.

La variable AGE pasará a ser una variable numérica implícita de 8 bytes cuando el compilador encuentre su nombre en la instrucción LABEL.

El tipo y longitud de una variable son muy importantes. El uso de longitudes muy largas para variables en archivos grandes desperdicia espacio de almacenaje y puede aumentar el tiempo de procesamiento.

Aunque el SAS le permite a un programador mezclar variables de caracteres y numéricas en una instrucción, el SAS se rige con reglas operacionales muy complicadas para tales casos y las complicaciones frecuentemente conducen a difíciles problemas de corrección de errores.

Recomendamos que todas las variables en un programa de entrada de datos se declaren con instrucciones LENGHT (y/o con instrucciones RETAIN, discutidos más adelante). En estos casos se omitieron variables del LENGTH para luego discutir los temas arriba abordados.

2.3.6 La instrucción LABEL

Aunque los nombres SAS para las variables son nemotécnicos y algo descriptivos, hay límites a la cantidad de información que se puede embutir en una nombre de 8 bytes. El SAS tiene para almacenamiento hasta 40 bytes de información adicional sobre cada variable en un "rótulo de variable". Los rótulos de variables son especificados en una o más instrucciones LABEL. (Ver SAS79[6], página 112 para detalles). La información en un rótulo se da solamente una vez. La información se almacena en el archivo, junto con el nombre de la variable, su longitud, otros. Esta información se usará en cualesquiera pasos de procesamiento futuro. Muchos de los análisis PROCs imprimen el rótulo junto con el nombre de la variable cuando el rótulo está presente.

El preparar un rótulo descriptivo para cada variable no obvia es buena práctica de programación. Sólo un principiante no le sacaría ventaja a esta característica de documentación. Se omitieron los rótulos de algunas variables en el ejemplo para hacer enfásis sobre este punto.

2.3.7 Variables DATE

Una de las mejores caracterísitcas del SAS79[6] es su habilidad para realizar computaciones sobre fechas automáticamente. Esta característica es extremadamente útil en el procesamiento de datos de investigación, como en el ejemplo, donde AGE es la diferencia entre dos fechas.

En el SAS79[6], una "Variable DATE" es simplemente una variable numérica cuyo valor representa un número de días (un número entero) desde

el 01 de enero de 1960 hasta una fecha dada. Puesto que una variable DATE es una variable numérica, no se requiere una aritmética especial. Sin embargo, el SAS provee funciones especiales para convertir las representaciones externas de fechas (e.g., 03Feb80) a la representación interna correcta, y viceversa.

El ejemplo tiene dos variables DATE, SURVDATE y DATEBORN, que son datos de entrada (véase la instrucción INPUT) de campos de 6 bytes en el formato yymmdd. En la alimentación de información, el SAS convierte los 6 dígitos (e.g. 800203 por 03Feb80) a un valor que representa el número de días desde 01ene60 a la fecha de los datos (e.g., a 03Feb80). Los años bisiestos y otras anomalías del calendario se toman en cuenta y se ajustan correctamente.

La diferencia entre dos variables DATE, e.g., SURVDATE-DATEBORN no es un valor DATE, porque no representa el número de días desde 01JAN60 hasta una fecha específica.

La diferencia de dos valores DATE sí es el número de días entre las fechas. Así, en el ejemplo, edad en años es computada como AGE= (SURDATE-DATEBORN)/365.25. El 0.25 extra en el denominador representa el hecho que un año es (hasta 5 dígitos) realmente 365.25 días, creando la necesidad de años bisiestos. Una discusión más detallada sobre varriables de fecha se encuentra más adelante.

2.3.8 La instrucción FORMAT

Como las variable DATE son simplemente variables numéricas con interpretación especial, SAS usará un formato estándar implícito numérico para imprimirlos, a menos que se le den instrucciones contrarias. Aunque puede ser interesante ver una fecha representada como el número de días desde 01 JAN 60, los seres humanos encuentran mucho más fácil otros formatos de fecha (e.g., 03FEB80, 800203, 80-02-03, y febrero 3, 1980).

La instrucción FORMAT permite al programador SAS asociar un formato de impresión específico con una variable. Entonces, cuando se imprimen los valores de la variable el SAS usará este formato, previamente especificado, para convertir de forma interna a forma externa.

(El programador puede cambiar el formato a la hora de usarlo). La información del formato se almacena en el archivo junto con el nombre de la variable, su longitud, su rótulo, etc. Un formato podría estar asociado con cualquiera de lod tipos de variables SAS.

Los formatos son tema para un estudio más a fondo, más adelante. Aquí basta decir que se usa la instrucción FORMAT para este propósito y que el ejemplo ilustra su uso directo. El formato de información producida en el ejemplo, tanto para DATEBORN como para SURVDATE, es de la forma yy-mm-dd, es decir, 8 bytes incluyendo guiones.

2.3.9 La instrucción INPUT

La instrucción INPUT es el núcleo del programa de la Fase de Entrada de Datos. La instrucción INPUT le especifica al SAS el nombre de cada variable, la localización de la variable en el registro de entrada de los datos, e información especial (el formato) para la conversión de formato externo a formato interno.

Esta sección contiene solamente información elemental sobre las instrucciones INPUT para datos de formato-fijo de una tarjeta por cada observación. La palabra clave INPUT es seguida por una "especificación" por cada campo de datos. En el ejemplo se colocan las especificaciones en líneas sucesivas a manera de que el informe tenga la apariencia de una lista, definiendo el formato de entrada de los datos. El SAS no requiere que la instrucción INPUT se registre de esta manera, pero es práctica de programación hacerlo así.

La primera especificación de campo, "CARDTYPE $1-5", tiene tres partes: el nombre de la variable (CARDTYPE), el símbolo "$", el cual significa que el campo es un grupo de caracteres (no número) y la especificación de las colomnas de la tarjeta para el campo, "1-5", indicando las columnas del 1 al 5. (El símbolo "-" se usa aquí como un guión y no como un signo de restar). Por cada tarjeta (y observación) esta especificación causará que el grupo de caracteres en las columnas 1 al 5 en la tarjeta de datos se traspasen a las localizaciones de la memoria, asociadas con CARDTYPE. Además, los primeros lugares en blanco, como

"bABCD" (en donde b denota el carácter en blanco), serán suprimidos por el desplazamiento de los caracteres no en blanco hacia la izquierda y la inserción de los espacios en blanco a la derecha. Así, la secuencia de caracteres de entrada "bABCE" se almacenará en el archivo SAS como "ABCD". Este proceso se llama "justificación izquierda" del campo de caracteres. (Este proceso puede ser invalidado con el formato $CHARw., discutido en las páginas 36-37 del SAS79[6]).

La segunda especificación, "IDNUM 6-9", es similar a la primera excepto que la omisión del "$" indica que el campo y la variable son numéricos. SAS convertirá el valor perforado en las columnas 6 al 9 a formato numérico interno. Se ignorarán los primeros y/o últimos blancos. (Esto es diferente de FORTRAM, por ejemplo, el cual maneja los últimos blancos como ceros). Si se perfora un decimal en el campo, el número se convertirá en la representación interna correcta, incluyendo la parte fraccionaria. La "notación E" también se convierte apropiadamente: "12E4" se convertirá en el equivalente interno de 120000, por ejemplo.

Los campos para NAME, SEX, HEIGHT y WEIGHT se procesan de manera similar al procesamiento de CARDTYPE e IDNUM.

2.3.10 Los formatos en las instrucciones INPUT

Los campos de datos, aparte de aquellos extrictamente numérica y de caracteres, requieren información especial adicional llamada "formato" para la adecuada conversión a formato interno. El SAS tiene una variedad sorprendente de formatos para tales casos especiales (SAS79[6], capítulo 5, páginas 29-44).

La instrucción INPUT del ejemplo en la Figura 2.2 contiene un ejemplo del uso de formatos especializados. Las variables DATEBORN y SURVDATE son variables de fecha SAS (ya discutidas en la sección 2.3.7). En una tarjeta de datos, cada una de estas variables ocupa 6 columnas en el formato yymmdd, en donde yy representa los últimos dos dígitos del año, mm representa el número del mes (01 para enero, 02 para febrero, etc.) y dd representa el día dentro del mes. Por ejemplo, 800203

para 03Feb80. A menos que hayan instrucciones de lo contrario, SAS manejaría estos datos como numéricos, con el valor 800203. Al ordenarle al SAS usar un formato de fecha (YYMMDD6 en este caso; véase Fig. 5) el programador le da instrucciones al SAS para que convierta el número de 6 dígitos en una de fecha variable que represente el número de días desde el 01JAN60. En el ejemplo, el valor "800203" de la tarjeta de datos, se convertiría a 7338, o sea, el número de días desde 01JAN80 al 02FEB80.

2.3.11 PROC PRINT

La instrucción "PROC PRINT" en el ejemplo (Fig. 5), ejecuta el procedimiento SAS PRINT. Nótese que este paso SAS tiene dos instrucciones, "PROC PRINT", y "TITLE3...". La instrucción TITLE3 agrega una tercera línea a la información del título generada por las instrucciones TITLE en el primer paso.

El procedimiento PRINT produce una impresión con un buen formato de contenido de un archivo SAS. Se puede especificar el nombre del archivo de entrada. Si fuese omitido se imprimen los contenidos del archivo creado más reciente (en este trabajo).

El procedimiento PRINT es muy sencillo y se describe en la página 353 del SAS79[6]. El lector debe leer esta descripción del PROC PRINT en este momento. Nótese que el ejemplo contiene otra ejecución de PROC PRINT después de que se ordenen los datos (PROC SORT).

La segunda ejecución de PRINT usa el parámetro ID, descrita en la p. 353 del SAS79[6].

2.3.12 Clasificación: Primer vistazo al PROC SORT

Clasificar un archivo SAS es realmente fácil. El ejemplo (Fig. 5) ilustra una simple ejecución de PROC SORT.

Generalmente se especificará un archivo SAS de entrada y otro de salida para la información producida por el procesamiento SORT, en la siguiente forma (Fig. 6).

Nótese el uso del parámetro DATA para especificar un archivo de entrada.

```
PROC SORT DATA=archivo de entrada SAS
          OUT=archivo de salida SAS;
          BY lista;
```

Fig. 6. Archivo de entrada y de salida

Esto contrasta con el uso de la instrucción DATA para especificar: (1) el comienzo de un paso DATA y (2) el nombre de un archivo de salida. El parámetro DATA (en una instrucción PROC) es completamente diferente de la instrucción DATA.

Si se omite el parámetro DATA de la instrucción PROC SORT (como en el ejemplo, SAS usará para fines de entrada el archivo SAS más reciente del trabajo.

Si se omite la especificación del archivo de salida (OUT=) en la instrucción PROC SORT, SAS hará que el nombre del archivo de salida sea el mismo del archivo de entrada.

Si el PROC SORT fallara por alguna razón extra, o termine antes de completar el archivo de salida, esto no afectaría al archivo de entrada.

Un paso PROC SORT siempre tiene que incluir un parámetro BY, el cual especifica las variables cuyos valores han de ser usados para determinar el ordenamiento de las observaciones. En el ejemplo se ordenan los datos con base en variable alfabética NAME, lo cual ordena las observaciones alfabéticamente, por los valores almacenados en NAME.

El lector ahora debe estudiar la discusión en SAS79[6] sobre el procedimiento SORT, páginas 373-375, excepto la sección "Clasificando grandes juegos de datos".

2.3.13 Resumen

Se ha presentado, por medio de un ejemplo, un programa "típico"

de la Fase de Entrada de Datos. La omisión principal es que el programa ejemplo no se revisa con el fin de buscar errores en los datos. Se discuten los instrumentos (instrucciones) SAS, usados para la detección de errores, más adelante.

Debe recordarse que el objetivo principal de un programa de Fase de Entrada de Datos SAS es el de alimentar datos dentro de un archivo SAS. Los archivos SAS son más fáciles de procesar, modificar y manipular que los archivos OS.

Si es usted un principiante con el SAS y el procesamiento de computadora, esta sección puede haberle sido poco familiar y probablemente no le haya encontrado sentido a algunas de las explicaciones. Principiante o no, es importante hacer el ejercicio en la siguiente sección, el cual puede suscitar importantes preguntas para su instructor o el experto SAS local. Los principiantes no deben preocuparse por su actual estado de confusión: hagan el ejercicio, relean este capítulo y entonces rehagan el ejercicio. Como se hizo notar en la introducción, el aprender a manejar datos es un proceso iterativo!

2.3.14 Ejercicios/Tarea

1. Léase cada sección del SAS79 a que se hizo referencia en la sección 2. Nótese que en algunos casos el leer un capítulo entero del SAS79 no se recomienda (porque el capítulo contiene descripciones de otras características aún no discutidas).
2. Hagan un archivo OS en disco para usar en ejemplos posteriores; probablemente necesitarán ayuda del instructor o experto SAS local.
3. Obténgase una copia de los datos descritos en el ejemplo (el formato se encuentra en el Cuadro No.). Hagan un programa SAS para entrar y clasificar los datos y almacenar los resultados en su base de datos SAS. (Siéntanse libres de copiar el programa en la Fig. 5; tal vez quisieran agregar rótulos a las variables "no rotuladas"). Agreguen un paso PROC CONTENTS al final del programa para imprimir información sobre todos los archivos en su base de datos.

Es muy probable que cometan errores; examinen cuidadosamente la sección LOG de la información producida (especialmente las "NOTES") para determinar si el programa ha funcionado bien.

2.4 Operadores números, Expresiones, Instrucciones y Funciones

Una de las mejores características del SAS es la facilidad con la que nuevas variables pueden ser creadas con base en las viejas variables. Una casi ilimitada variedad de transformaciones y combinaciones numéricas puede hacerse con expresiones, instrucciones y funciones numéricas.

2.4.1 Operadores Numéricos, Instrucciones de Asignación y Expresiones

Una instrucción numérica típica se expresa en el formato:

```
variable = expresión;
```

en donde variable es el nombre de una variable numérica y expresión es una expresión numérica SAS.

Las expresiones numéricas SAS son muy similares a las expresiones numéricas en otros lenguajes de computación tales como Fortram y PL/I. Se asume que el lector está algo familiarizado con tales expresiones en otros lenguajes.

Los operadores numéricos SAS (definidos en el SAS79[6], Apéndice 5) son los que se indican en el Cuadro No. 3.

Cuadro No. 3 Operadores numéricos SAS

Operador	Nombre	Código SAS	Matemáticas
**	Exponencial	A**B	A^B
*	Multiplicación	C*D	CXD
/	División	E/F	E/F
+	Suma	G+H	G+H
-	Resta	I-J	I-J
> <	Mínimo	K> <L	Min (K,L)
< >	Máximo	M< >N	MAX (M,N)

Se combina las constantes, variables y operadores para formar expresiones. En el ejemplo anterior tenemos la instrucción:

AGE = (SURVDATE-DATEBORN)/365.25

que convierte la edad en días a edad en años.

Las constantes numéricas, tales como 365.25 son generalmente codificadas apenas aparecen. Para formas especiales (Notación-E, hexadecimal), véase Apéndice 5 del SAS79[6].

"Precedencia de los Operadores". Uno puede usar paréntesis para definir el orden de varias operaciones, como en la expresión de arriba. En algunos casos si el paréntesis no define el orden de ejecución de las operaciones:

a. Los operadores con alta precedencia se ejecutan antes que los operadores con baja precedencia.
b. Los operadores con igual precedencia se ejecutan en orden de izquierda a derecha, excepto en el primer grupo. (Véase el Apéndice 5. En caso de duda use paréntesis).

La tabla de precedencia de operador es la siguiente:

El más alto	**, prefijo-, prefijo+, NOT, > <, < >
	*,/
	Operadores de comparación (<,<=, etc.)
	AND
El más bajo	OR

Así por ejemplo, se podría codificar

$$A = \frac{c}{d} + \frac{e}{f + g} + \frac{h + i}{j}$$

como

A = C/D+E/(F+G) + (H+I)/J

Aquí se requieren ambos juegos de paréntesis. Similarmente,

$x = \sqrt{y^2 + z^2}$ podría ser codificado

como

```
X = SQRT(Y*Y+Z*Z)
```

o

```
X = SQRT(Y**2+Z**2)
```

o

```
X = (Y**2+Z**2)**0.5
```

En este ejemplo, el primer código es preferible porque Y**2 se valúa como Exp (2*Log (Y))- Generalmente, A**B se evalúa como Exp (B*Log (A)). Esta expresión falla y crea mensajes de error si A < 0. Así, los poderes de números enteros de bajo orden, como Y^2, Y^3, Y^4, se codifica mejor como una multiplicación repetida: Y**Y, Y*Y*Y, etc.

2.4.2 Funciones aritméticas

El SAS provee una amplia variedad de funciones aritméticas, descritas en el Apéndice 1 del SAS79[6]. Las funciones aritméticas son:

ABS	valor absoluto
CEIL	entero más pequeño > argumento
FLOOR	entero más grande < argumento
INT	truncamiento del argumento a entero
MOD	Mod (a, b) = a mod b
SQRT	raíz cuadrada del argumento
ROUND	redondeo al entero siguiente
SIGN	=+1 o -1 dependiendo del signo del argumento

(SAS también incluye MIN y MAX entre las funciones aritméticas pero éstas, en realidad, son funciones estadísticas).

Funciones Trigonométricas e Hiperbólicas. El SAS provee todas las funciones trigonométricas e hiperbólicas usuales: SIN, COS, TAN, ARCOS (arco coseno), ARSIN (arco seno), ARTAN, COSH, SINH, TANH.

Funciones matemáticas. El SAS provee una útil colección de funciones matemáticas y matemático-estadísticas.

LOG, LOG10, LOG2	Logaritmos a la base e, 10 y 2, respectivamente
GAMMA, LGAMMA	Función gamma y función log gamma, respectivamente

GAMINV — Función gamma inversa

EXP — Función exponencial: e^x

PROBNORM, PROBCHI, PROBT, PROBF, PROBGAM — Probabilidades computadas de la distribución indicada (normal, chi-cuadrada, t, f, Gamma).

Generadores de números al azar. SAS provee UNIFORM y NORMAL para generar números al azar de las distribuciones uniformes estándar (0, 1) y normal estándar N(0, 1).

2.4.3 Funciones estadísticas

SAS provee 15 funciones para computar estadísticas de los argumentos. Estas funciones son distintivas en cuanto:

a. El número de argumentos no es fijo - se puede usar cualquier número de argumentos, por ejemplo:

```
X = MEAN (Y1, Y2, Y3); Y = MEAN (Y1, Y2, Y3, Y4);
```

b. Se ignora cualquier argumento que tiene un valor omitido en las computaciones (excepto las funciones N, NMISS).

c. Una forma especial de argumento puede usarse para listas largas de variables.

La lista y definiciones de funciones estadísticas se encuentran en la página 444 del SAS 79[6]. Estas funciones son: N, NMISS, SUM, MEAN, MIN, MAX, RANGE y otras. En "uso estándar" los argumentos se separan con comas:

```
X = MIN (3, X2, X3, X7);
```

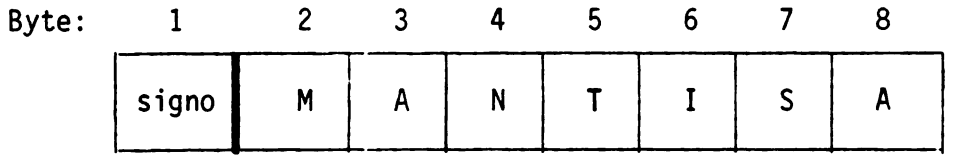
Para estas funciones estadísticas solamente, listas de variables cuyos nombres terminan en números pueden escribirse en la siguiente forma:

```
Y = MAX (OF X1 - X100);
```

lo cual tiene el mismo efecto de escribir todo de X1, X2,...X100, separados por comas. Como se notó antes, estas funciones estadísticas ignoran los valores omitidos en las computaciones, excepto en el caso de las funciones N y NMISS. N cuenta el número de argumentos con valores no omitidos y NMISS cuenta el número de argumentos con valores omitidos.

2.4.4 Valores omitidos

Generalmente, en la alimentación de información si un campo de datos para una variable numérica está en blanco, el SAS almacenará un "valor omitido" especial para esa variable en esa observación. Normalmente se almacena un valor numérico en un formato de punto flotante hexadecimal de 8 bytes, como sigue:



Mantisa 7-bytes (14 números hexadecimales)

Signo (primer bit) y exponentes de base 16 (7 bits)

Si el valor de los datos es cero, SAS guarda una mantisa cero, a + signo, y un valor exponente de 0 (para 16**0). Si el valor de los datos está omitido, SAS guarda una mantisa cero y una secuencia binaria 1 000 0000 (hex 80) en el byte del signo/exponente. En los procesamientos posteriores, antes de realizar las operaciones aritméticas SAS verifica cada valor numérico para ver si es un "valor omitido". Si así es, el resultado de la operación es un valor omitido.

Un programador puede codificar valores omitidos con el carácter ".", en una expresión numérica SAS, por ejemplo:

```
X = .;
```

colocará un valor omitido en X.

En realidad SAS, permite 28 tipos de valores omitidos, los cuales pueden codificarse como sigue: ".", "._", ".A", ".B", ".C",..., ".Z". Para "._" SAS almacena un carácter "_"; para ".A" hasta ".Z", SAS almacena el carácter "A",..., "Z" en el primer byte.

El valor omitido "." es llamado el "valor omitido simple". Normalmente se coloca en una variable numérica cuando el campo de alimentación está en blanco. Los otros valores omitidos se llaman "valores omitidos especiales" y se pueden usar en instrucciones de programas, tal como X = .A; estos también pueden ser generados por SAS mediante uso de la instrucción MISSING.

Esto está muy bien descrito en el capítulo 6 de Valores Omitidos, SAS79[6], que debe ser leído en este momento.

2.4.5 La instrucción RETAIN

En operación "normal", antes de empezar el proceso inicial y después de haber producido una observación para un archivo SAS, el SAS ajusta todas las variables de carácter a blanco.

En este modo de operación el programador no puede tener acceso a ninguna información de la observación precedente. Por ejemplo, supóngase que se desea contar el número de observaciones. Un programador con experiencia en FORTRAM o PL/I podría probar el siguiente programa:

```
    N=0;
DATA nombre;
    ...
    INPUT lista;
    N= N+1;
```

Este programa no funciona. El SAS no acepta la instrucción "N=0;" fuera de un paso DATA. Dentro del paso DATA, el resultado de la instrucción "N=N+1;" (asumiendo que N no está en la lista de INPUT) sería el ajustar N a un valor omitido, porque N se ajustaría a un valor omitido antes de cada ejecución de las instrucciones en el programa.

La instrucción RETAIN le permite a un programador especificar variables que no ajustarán a valores omitidos o en blanco antes de la ejecución de las instrucciones en el paso. El programadcr también puede especificar un valor inicial (constante) diferente de omitido (".") para una variable con la instrucción RETAIN.

El ejemplo anterior podría hacerse con el siguiente programa:

```
DATA nombre;
     RETAIN N (0);
     INPUT lista;
      N = N+1;
      (otras instrucciones)
```

Aquí N es especificado como una variable cuyo valor no se cambiará después de que SAS produzca cada observación. Aún más, N se ajusta inicialmente a cero. Así la instrucción "N = N + 1;" siguiendo a la instrucción INPUT, efectivamente cuenta el número de veces que se ejecuta el INPUT.

Se describe el informe RETAIN en las páginas 107-109 del SAS79[6], y debe leerse en este momento.

El lector debe estar enterado de que SAS tiene otros medios para lograr el conteo de registros (como arriba) y otras tareas similares. La instrucción "SUM" (p. 103) puede ser usada, por ejemplo, o la variable automática _N_, la cual sirve un propósito similar. (Ver las descripciones de las funciones LAG y DIFF, en el SAS79[6] p. 440, para computaciones LAG y relacionadas). Deben leerse secciones del SAS79[6] que describen estas características cuando surge la necesidad de usarlas.

2.5 Manipulación de Caracteres

SAS ofrece facilidades algo limitadas pero muy útiles para almacenar y manipular variables carácter, i.e., variables cuyos valores son secuencias del tipo carácter EBCDIC en vez de números para fines de computación. Desafortunadamente, muy pocas de estas facilidades se describen en SAS79[6].

2.5.1 Variables de Carácteres

SAS tiene dos tipos de variables: (1) variables numéricas, cuyos valores están almacenados en un formato interno especial conveniente para computación numérica, y (2) variables carácter cuyos valores tienen el mismo formato interno y externo (EBCDIC). Los valores carácter, o "secuencias", no pueden usarse directamente para computación. Si la secuencia de caracteres contiene un número en el formato EBCDIC, la secuencia puede ser convertida a formato numérico interno para uso en computaciones numéricas. El valor también puede mantenerse en formato EBCDIC y manipulado como una secuencia carácter.

Cada carácter en una secuencia de caracteres requiere un byte para almacenaje. Un carácter blanco ("b") es solamente uno de los caracteres disponibles y por consiguiente también requiere un byte para almacenaje. (Se usará "b" para denotar el carácter blanco).

Una variable carácter SAS tiene un largo fijo que es el número de bytes disponible para almacenar valores de esa variable.

2.5.2 Determinando el Tipo y Largo de las Variables Carácter

En un paso DATA...INPUT, el compilador SAS determina el tipo de una variable (carácter o numérica) y el largo (número de bytes asignados para almacenar los valores desde la primera aparición del nombre de la variable en el programa SAS. Si una variable no se especifica como variable carácter, SAS asumirá que la variable es numérica. SAS tratará de determinar el largo debido, para una variable carácter, de la información disponible. Si la primera aparición de la variable carácter es en una instrucción INPUT, SAS determinará el largo basándose en esta instrucción. En el ejemplo anterior, al crear el archivo PFD79, el nombre de variable CARDTYPE aparece primero en la instrucción.

```
INPUT CARDTYPE  $ 1 - 2
                etc.;
```

El "$" indica que CARDTYPE debe ser una variable carácter. SAS determina de las columnas de alimentación especificadas (1-2) que la longitud de la variable debe ser de dos bytes.

SAS también puede determinar el tipo y largo de una variable carácter desde su primer aparición en una instrucción de asignación. Si la instrucción INPUT de arriba fuera seguido por A = CARDTYPE; entonces SAS definiría la letra A como una variable carácter con el mismo largo de CARDTYPE.

El programador puede especificar el tipo y largo de variables carácter (y numéricas) en instrucciones LENGTH, como se describió anteriormente. Una instrucción LENGHT es una declaración, o definición, y no es ejecutable. Una vez que el tipo y largo de una variable se establecen no se pueden cambiar fácilmente. Se describe la instrucción LENGTH en la página 54 del manual SAS79[6] y está ilustrado en el ejemplo PFD79.

La instrucción LENGHT tiene que seguir a la instrucción DATA y preceder cualesquiera otras instrucciones conteniendo nombres de las variables en la instrucción LENGHT.

Es una buena práctica de programación el especificar explícitamente el tipo y longitud de las variables carácter en las instrucciones LENGHT.

2.5.3 "Valores Omitidos" para las Variables Carácter

Puesto que las variables carácter no pueden participar directamente en computaciones numéricas, no hay problema de "valores omitidos" para las variables carácter. Cualquier carácter válido EBCDIC en los datos de alimentación puede ser alimentado a una variable carácter. Un campo de alimentación blanco da lugar a que la variable carácter contenga caracteres blancos.

2.5.4 Conversión entre Valores Carácter y Numéricos

El compilador SAS intentará una conversión automática de un valor carácter a un valor numérico, o viceversa, cuando valores carácter y numéricos están mezclados en la misma instrucción. Por ejemplo, si X es una variable numérica y C es una variable carácter con un largo 8 bytes, la instrucción

```
C = X;
```

será compilada a manera de que el valor numérico en X sea convertido a una secuencia carácter EBCDIC de 12 bytes (usando el formato BEST12); los 4 caracteres más a la derecha se suprimirían y los 8 caracteres más a la izquierda se almacenarían en C. (ver página 11 del manual SAS79[6] para más detalles). Esto puede o no producir el resultado que el programador quiere.

La instrucción,

```
X = C;
```

se comilaría de manera que SAS intentaría convertir los 8 caracteres EBCDIC en C a formato numérico interno. Si la conversión es exitosa, el número resultante se almacena en X. Si la conversión no es exitosa (e.g., si C = "ABCDEFGH"), se coloca un valor omitido simple (.) en X.

SAS intentará conversiones siempre que sean necesarias. Por ejemplo, usando el C de arriba, la instrucción "X = SQRT (C);" tendría esencialmente el mismo resultado de las dos instrucciones.

```
TEMP = C;
X = SQRT (TEMP);
```

en donde TEMP es una variable numérica.

Las dos funciones tipo carácter PUT e INPUT (SAS79[6], p. 439) están disponibles para el programador para realizar transformaciones

explícitas entre valores numéricos y de carácter. Se pueden obtener de modo confiable los resultados requeridos usando estas funciones. El dejar que SAS realice las conversiones implícitamente no es buena práctica de programación.

2.5.5 Comparación de Valores Carácter

Los operadores de comparación SAS (<, < =, =, =>, >, r=) pueden ser usados para comparar valores carácter. El resultado de tal comparación es siempre un valor "lógico" (i.e., un valor numérico con 1 representando "Verdadero" y 0 representando "Falso"). (La comparación de valores carácter se discute en detalle en el Apéndice 5 del SAS79[6], pp. 458-459).

Si dos valores carácter de diferentes longitudes han de ser comparados, SAS moverá el valor menor a un almacenaje temporal, agregará blancos a la derecha hasta que el valor temporal sea del mismo largo del otro valor y luego hará la comparación entre las dos secuencias de igual largo.

Con el propósito de determinar el ordenamiento de caracteres (e.g., ¿es "%" menor que "$"?), el SAS usa la secuencia de comparación" estándar IBM 360/370. El orden de todos los caracteres comunes se encuentra en la p. 458 del SAS79[6]. En esta secuencia, los grupos de caracteres "imprimibles" son:

(blanco) < caracteres especiales < letras < números

Los caracteres especiales son tales como: . =) +, etc. Las letras son A < B < --- < Z y los numerales (los códigos EBCDIC para dígitos) son 0 < 1 < 2 < --- <9.

2.5.6 Concatenación

El operador SAS que "combina" dos valores carácter para producir otro valor carácter es el operador de concatenación, ||, descrito en la p. 459 del SAS79[6]. (El lector debe estudiar esta descripción en el SAS79[6] antes de proceder).

La concatenación de nombres puede ser un poco difícil porque el operador || recorta (suprime) blanco antes de concatenar. Por ejemplo, el siguiente código:

```
LASTNAME = ' PAEZb';
FIRSTN   = ' GILBERTOb';
NAME     = LASTNAME||FIRSTN;
```

Producirá los mismos resultados que:

```
NAME     =  'PAEZGILBERTO';
```

Una manera de resolver el problema es insertar un blanco o coma:

```
NAME     =  LASTNAME||'b'||FIRSTN;
         o
NAME     =  LASTNAME||','||FIRSTN;
```

Esta discusión se refiere a la versión 79.2 y versiones previas de SAS. En la versión SAS79.3 y futuras versiones, el operador concatenador sencillamente concatena las dos secuencias de caracteres.

2.5.7 Funciones Carácter

Las funciones suministradas por SAS para la manipulación de valores carácter se describen en la p. 439 del SAS79[6]:

SUBSTR para extraer una subsecuencia de caracteres;

INPUT para transformar explícitamente una secuencia EBCDIC a formato interno, usando un formato SAS;

PUT para transformar explícitamente un valor interno (formato de secuencia numérica o carácter, usando un formato SAS);

LENGTH para permitir al programa determinar el largo de una secuencia carácter, omitiendo los blancos de la derecha;

REVERSE para reversar el orden de caracteres en una secuencia. La revisión 79.3 del SAS contiene una gran variedad de funciones para manipular secuencias de caracteres; se describirán estas funciones con más detalles en una versión futura del manual.

Los detalles técnicos de estas funciones son dados en la p. 439 del SAS79[6].

2.5.8 Formatos Carácter

SAS provee los formatos carácter necesarios, para el uso en INPUT, PUT e instrucciones relacionadas con ellos (y las funciones INPUT y PUT) para la transformación de caracteres de almacenamiento externo a o desde almacenamiento interno. Se describen formatos $w., $CHARw., $HEXw., y $VARYINGw., en las pp. 36-37 del SAS79[6]. Hay muy poca diferencia entre almacenaje interno y externo de secuencia carácter. La principal diferencia es en el manejo de blancos iniciales y la longitud.

Si una secuencia de entrada es muy larga para la variable carácter, la secuencia es truncada a la derecha, para que quepa. Se suprimen caracteres más hacia la derecha hasta que la secuencia es lo suficientemente corta para que quepa. Si una secuencia de entrada es más corta que la variable carácter, se rellena la secuencia a la derecha con blancos. (Se agregan blancos a la derecha hasta que la secuencia es del mismo largo de la variable).

El formato "$w." causa que los primeros blancos sean suprimidos de una secuencia de entrada, el primer carácter del resultado almacenado no será en blanco a menos que el valor completo sea blanco. El formato "$CHARw." no suprime los primeros blancos de entrada.

El formato "$VARYING." es muy útil como "formato libre" para la entrada de datos carácter. El uso de este formato se describirá subsecuentemente.

2.6 Instrucciones Lógicas: IF, THEN, ELSE, GO TO, etc.

2.6.1 Valores Lógicos (Booleano), Variables y Expresiones

Las variables lógicas (Booleanas) tiene solamente dos posibles valores, verdadero y falso. Por conveniencia SAS (y muchos otros lenguajes de programación) codifican estos valores numéricamente: 1 para VERDADERO, 0 para FALSO. Así, en SAS, una "variable lógica" es realmente sólo una variable numérica; el programador fuerza la variable a ser Booleana al restringirla a los valores 0 y 1.

Se construyen las expresiones lógicas (booleanas) a partir de los operadores Booleanos AND (&), OR (I), y NOT (¬), y de comparaciones. (Ver SAS79[6], Apéndice 5, para más detalles).

Las comparaciones son expresiones de la forma: [(variables (o valor)) (operador de comparación) (variable (o valor))] como por ejemplo,

```
A < B
NAME  =  'JONES'
```

En SAS, las comparaciones siempre producen un resultado de 0 ó 1.

La comparaciones pueden ser combinadas por operadores Booleanos para producir expresiones Booleanas como, por ejemplo,

```
(A < B)  AND (NAME = 'JONES')
```

Los paréntesis se usan aquí para hacer explícito el orden de evaluación. El valor de una comparación o expresión Booleana puede ordenarse en una variable numérica.

```
X  =  (A < B) & (NAME = 'JONES');
```

Se puede también realizar computaciones numéricas sobre valores Booleanos:

```
NUM = (X1<5) + (X2<5) + X3<5);
```

Aquí, NUM es el número de variables (entre X1, X2, X3, X4) que exceden 5. El principal uso de las expresiones Booleanas está en las instrucciones IF, descritas más adelante.

2.6.2 Rótulos de Instrucciones y GO TO

Dentro de cada ejecución de un paso SAS, las instrucciones SAS "normalmente" se ejecutan en el orden en que aparecen en el programa. El programador usa rótulos de instrucciones y la instrucción GO TO para ejecutar las instrucciones en un orden diferente.

Un rótulo de instrucción es un nombre exclusivamente del SAS al comienzo de una instrucción, seguido de dos puntos, como por ejemplo:

```
TOP:  X = X + 1;
```

El rótulo TOP identifica la instrucción.

La instrucción GO TO rótulo traslada el control a la instrucción indicada por el rótulo. Esto puede ser un salto "hacia adelante" o "hacia atrás" como se ilustra en el segmento del programa:

```
STEPA:  A = B+C;
        GO TO STEP 2;
STEP1:  A = D+B;
        GO TO STEP A;
STEP2:  Y = SQRT (Z);
          ...
```

Aquí, el primer GO TO salta hacia adelante, el segundo salta hacia atrás. Más detalles sobre rótulos de instrucciones y GO TO se encuentran en las pp. 113-114 del SAS79[6].

2.6.3 Grupos DO-END

Algunas veces, como en las instrucciones IF discutidas más adelante, es conveniente tratar un grupo de instrucciones como un bloque, casi como una instrucción. Tal grupo de instrucciones es especificado con una instrucción "DO;" inicialmente, y con una instrucción terminal "END;" como por ejemplo:

```
TOP:  DO;
         X = X+1
         Z = SQRT (X);
         T = X * Z;
      END;
```

El usar sangrías en las instrucciones en un "grupo DO" no es requerido por SAS, (pero es una buena práctica de programación). SAS trata todas las instrucciones en un grupo DO-END como un solo bloque, como se verá en la siguiente sección.

2.6.4 IF-THEN y IF-THEN-ELSE

Las instrucciones de control más importantes en SAS son las instrucciones IF. Una completa explicación de las instrucciones IF requiere varios capítulos de un libro sobre programación, mucho más que los pocos párrafos disponibles aquí. (Una muy breve descripción de instrucciones IF-THEN se encuentra en la p. 113 del SAS79[6]. El lector debe leer esta descripción en este momento).

El uso principal de las instrucciones IF en MDI es para detectar errores de datos, tal como verificar si los valores de una variable se encuentran dentro de un rango razonable:

```
ERR = 0;
IF AGE<=0 OR AGE>=80
     THEN DO;
             PUT/'AGE OUT OF RANGE';
             ERR = 1;
          END;
```

o si una variable tiene un valor válido:

```
ERR = o;
IF SEX NE 'M' & SEX NE 'F'
   THEN DO;
          PUT/'INVALID VALUE FOR SEX';
          ERR = 1;
        END;
```

El otro uso principal es el ejecutar selectivamente ciertas instrucciones como:

```
IF ERR THEN LIST;
```

Esta instrucción, siguiendo las anteriores, imprimiría en el LOG los valores de todas las variables cuando se haya descubierto un error (cuando ERR = 1). Nótese que son limitados los tipos de instrucción que pueden seguir a THEN o ELSE (SAS79[6], p. 113); en particular, un IF no puede seguir a THEN o ELSE. Puede usarse DO para resolver el problema:

```
ERR=0
IF SEX = 'M'
  THEN DO;
          IF HEIGHT>210 THEN ERR=1;
       END;
  ELSE DO;
          IF HEIGHT>190 THEN ERR=1;
       END;
```

Noten que el código anterior podría ser abreviado usando ERR como una variable Booleana:

```
ERR=0;
IF SEX = 'M' THEN ERR = ERR OR HEIGHT>210;
             ELSE ERR = ERR OR HEIGHT>190;
```

2.6.5 ERROR, STOP, y ABORT

Algunas veces se suscitan condiciones de errores y el programador quiere que su programa le señale la condición y/o terminar el proceso. Las instrucciones ERROR, STOP y ABORT proveen tres niveles de acción para tales condiciones. Estas instrucciones están descritas en las pp. 106-107 del SAS79[6].

Brevemente, "ERROR;" o "Mensaje ERROR;" causa que el indicador de error del sistema para la observación actual, _ERROR_, sea ajustado a '1'. Si un mensaje está presente, se imprime. Una vez que se ha completado el procesamiento de la observación, si _ERROR_= 1, se imprimen los valores de todas las variables en el LOG (como si LIST fuese ejecutada).

La ejecución de STOP termina la ejecución del paso SAS actual. SAS irá al siguiente paso SAS (DATA o PROC) e intentará continuar.

La ejecución de ABORT termina el procesamiento SAS, devolviendo el control al sistema operativo. ABORT también permite al usuario ABEND el trabajo con un código de terminación del usuario. (Veáse SAS79[6] p. 107 para más detalles).

2.7 Arreglos y Anillos (arrays and loops)

2.7.1 Arreglos

En el SAS79[6] un arreglo es una colección de variables a ser procesadas juntas en pasos de DATA. La ventaja de un arreglo es el hecho de que se puede usar un solo nombre (el nombre del arreglo) para representar cualquier variable en el arreglo.

Todas las variables en un arreglo tienen que ser del mismo tipo (carácter o numérica). Las longitudes de las variables pueden variar.

Un arreglo es definido en la instrucción ARRAY (una declaración, una instrucción no ejecutable) la cual tiene la forma general (SAS79[6], pp. 12-13): ARRAY nombre del arreglo (nombre del índice), elementos del arreglo;
El ejemplo en SAS79[6] es

```
ARRAY Q (I) Q1-Q20;
```

El nombre del arreglo es Q (un nombre único dentro del paso DATA). El índice del arreglo es I. Las variables Q1, Q2, ..., Q20 son elementos del arreglo.

Siempre que el índice del arreglo, I, tiene un valor, 1<= I<= 20, el nombre de la variable Q se refiere al elemento Iavo del arreglo. Por ejemplo, cuando I=1, Q es lo mismo que Q1, cuando I=2, Q es lo mismo que Q2, etc.

Implementación:

Cuando el compilador SAS encuentra una instrucción ARRAY, prepara una lista de las variables en el arreglo y marca el nombre del arreglo como un tipo especial de variable. Cuando el nombre del arreglo se encuentra en una instrucción subsecuente, el compilador crea un código especial. Cuando se ejecuta este código, se determina el valor de la variable índice ("I" en el ejemplo), encuentra la dirección como la dirección efectiva. (Esto se llama una "implementación del indicador" porque el índice del arreglo es en realidad una variable indicadora). La implementación se asemeja fuertemente al uso de las variables indicadoras PL/I y es muy diferente de la implementación de los arreglos en FORTRAN o PL/I.

Los arreglos tienen una amplia variedad de usos en la programación MDI, algunos de los cuales se ilustraran en una sección subsecuente.

2.7.2 Listas de Variables

Los programas SAS79[6] típicamente contienen un número de listas de nombres de variables, o "listas de variables". SAS provee varios métodos para abreviar ciertos tipos de listas de variables. (Véase SAS79[6], p. 9).

La forma más elemental es aplicable a "nombres numerados", nombres de variable de la forma "aaaannn", en donde "a" representa una letra (o subrayado) y "nnn" representa un número sin ceros primeros. Los ejemplos (del SAS79[6]) incluyen

Q2 Q3 Q4 Q5 SEX1 SEX2 SEX3

etc. Una lista tal como la anterior puede ser abreviada en la forma

aaaannn - aaaammm

en donde "mm" es un número mayor que nnn. La lista anterior se vuelve

Q2 - Q5 and SEX1 - SEX3. Todas las variables intermedias implícitas tienen que existir, e.g., en Q2 - Q5, Q3 y Q4 tienen que existir.

SAS también permite especificaciones de "alcances de variables", pero esta técnica requiere conocimiento exacto del orden en el cual SAS está almacenando variables en una observación. No se recomienda esta técnica para programas de calidad profesional porque subsecuentes modificaciones de alguna parte del programa, que alteran el orden en el cual las variables están almacenadas, probablemente introducirán errores en la instrucción que se esté usando con un alcance de especificación de variables.

La abreviatura _NUMERIC_ como una lista de variables se traduce como a una lista que incluye todas las variables numéricas definidas a ese punto del programa. Las abreviaturas _CHARACTER_ y _ALL_ producen resultados similares para las variables carácter y todas las variables (numéricas y carácter) definidas a ese punto del programa.

Téngase cuidado con las listas de variables. En caso de duda hágase una lista exacta conteniendo cada nombre deseado. Si una lista es usada repetidamente, un MACRO puede ahorrar trabajo.

Las listas de variables tienen una aplicación obvia en las instrucciones ARRAY.

2.7.3 Anillo DO

Los anillos DO están entre las más poderosas de todas las instrucciones de control. Los anillos, junto con los arreglos, proveen capacidades de programación que son extremadamente difíciles de programar con otras combinaciones de instrucciones. (Las instrucciones DO se describen en las pp. 115-116 del SAS79[6]).

DO iterativo. La instrucción DO iterativo tiene una de dos formas generales,

```
DO índice = inicio TO alto;
DO índice = inicio TO alto BY incremento;
```

Por ejemplo, considérese la parte siguiente de un programa, en el cual se asume que X1, X2,..., X100 tienen valores (por INPUT u otro modo) y de desea contar el número de Xi que tienen valor absoluto mayor que 0.5.

```
ARRAY X (I) X1-X100;
NUM = 0;
DO I = 1 TO 100;
   IF ABS (X)>0.5 THEN NUM = NUM + 1;
END;
```

Se ejecutará la instrucción IF 100 veces; en cada ejecución X representará una variable sucesiva, X1, X2, ..., X100.

El DO del SAS79[6] es muy similar al DO del PL/I; el lector debe examinar la documentación en el SAS79[6] (pp. 115-116).

DO OVER. La instrucción de arreglo se compara con el DO iterativo en la instrucción DO OVER, descrito en la p. 116 del SAS79[6]. Esta forma del DO es pespecialmente útil para programas generales o "subprogramas" para problemas de MDI.

2.8 Macros y Subrutinas

SAS provee instrucciones Macro y capacidades de subrutina que permiten al programador usas de nuevo grupos de instrucciones de programa.

2.8.1 Macros

El propósito de una instrucción MACRO es el de modificar el programa SAS actual mediante la repetida inserción de texto de progrma. Un uso típico ocurre cuando un programador tiene una lista larga de variables que deben ser usadas en varios lugares en un progrma y que no pueden ser abreviadas por algunas de las técnicas ya discutidas. Ilustraremos una corta lista de variables para reducir el tamaño del ejemplo mostrado en la Figura 7.

```
//SYSIN DD *
   MACRO IDLIST PROVINCE CANTON DISTRICT IDNUM %
   MACRO VARLIST IDLIST FIELDNUM
                 CROP ALTITUDE SOILTYPE
                 RAIN_JAN RAIN_FEB RAIN_MAR %
   DATA DATABASE.FARM;
      (varias instrucciones del paso DATA)
   PROC PRINT;
      ID IDLIST;
      VAR VARLIST;
      TITLE DATOS DE ENTRADA EN EL ORDEN ORIGINAL;
  PROC SORT; BY IDLIST;
  PROC PRINT;
     ID IDLIST;
     VAR VARLIST;
     TITLE DATOS DE ENTRADA ORDENADOS PRO IDLIST;
```

Fig. 7. Ejemplo del uso de las instrucciones MACRO.

```
El Primer Paso PROC PRINT después de la sustitución del texto
por MACRO se convierte en:
PROC PRINT:
          ID PROVINCE CANTON DISTRICT IDNUM;
          VAR PROVINCE CANTON DISTRICT IDNUM
          FIELDNUM CROP ALTITUDE SOILTYPE
          RAIN_JAN RAIN_FEB RAIN MAR;
          TITLE DATOS DE ENTRADA EN EL ORDEN ORIGINAL;
```

Fig. 8. Efecto de las instrucciones MACRO en la Fig. 7.

(las otras sustituciones se dejan como ejercicios).

Cuando el compilador SAS encuentra el nombre de un MACRO (VARLIST o IDLIST en el ejemplo), reemplaza el nombre del MACRO por el texto del macro, i.e., por todos los caracteres entre el nombre MACRO y el símbolo "%". (Ver SAS79[6], p. 12, para una breve descripción de MACRO).

En este ejemplo el compilador modificaría el primer paso PROC PRINT de la Fig. 2 para aparecer como en la primera sección de la Fig.2. Se han reemplazado las palabras VARLIST y IDLIST por el texto de los MACROS. Nótese como el texto de IDLIST es sustituido en VARLIST como una parte de la 'expansión'.

SAS no imprime automáticamente el texto del programa después de la sustitución. Se puede especificar que las expansiones MACRO se imprimen, colocando una instrucción "OPTIONS MACROGEN;" en cualquier sitio anterior a la invocación de la opción. (Ver SAS79[6], p. 446). Ejercicio: Complete la Fig. 8, es decir, escriba la expansión completa del texto de la Fig. 7 como se produciría con el SAS.

2.8.2 Subrutinas

Las instrucciones MACRO de SAS permiten al programador insertar el mismo texto muchas veces en su programa. Si el MACRO contiene varias instrucciones (sin rótulos de instrucción), el texto del programa original será modificado por el compilador para que, en efecto, las mismas instrucciones sean insertadas en diferentes lugares y pueden ejecutarse varias veces.

Una subrutina, o subprograma, es un juego de instrucciones que aparece solamente una vez en un paso SAS pero que puede ser ejecutado varias veces desde diferentes lugares en el paso. La función SORT es un ejemplo de una subrutina incorporada por SAS. Cuando el compilador encuentra un SQRT en el programa produce un código para transferir el control a la llamada de subrutina SQRT.

En el SAS79[6] se usan las instrucciones LINK y RETURN para programar las subrutinas. (Véase SAS79[6], p. 114, para una breve descripción de estas instrucciones). Como simple ilustración, supóngase que unas

computaciones complicadas han de realizarse en diferentes lugares en un programa, involucrando una variable X, una M media y una desviación estandar S. La codificación de la Fig. 9 podría usarse para este propósito.

El primer GO TO es necesario para que la primera subrutina no se ejecute al principio del procesamiento de cada observación.

El "nombre" de la subrutina es COMP, el cual es precisamente el rótulo de la primera instrucción en la subrutina. Nótese que la subrutina es llamada por una instrucción "LINK COMP;", más adelante.

Antes de una instrucción "LINK COMP;", se colocan valores apropiados en X, M y S o sea los argumentos de entrada de la subrutina. Después del LINK se mueve el resultado desde RESULT a una variable apropiada. (AGE_F en el primer ejemplo).

Nótese que las instrucciones en COMP tienen acceso a todas las variables actuales. A menos que sea suprimida, cualquier variable que se use en COMP será parte de la salida. Las subrutinas SAS son "subrutinas internas", a diferencia de las subrutinas FORTRAN, por ejemplo.

Una subrutina en un paso SAS no está disponible para otros pasos SAS. Si se necesita en varios pasos SAS, debe convertirse en un MACRO e insertarse en cada paso por medio de sustitución MACRO. La subrutina está dentro del MACRO:

```
MACRO  COMP_MAC
       COMP:
       RETURN; %
```

En cada paso en que se incluye, se usaría "LINK COMP;" para involucrar la subrutina.

2.8.3 Otros Usos de RETURN

Nótese que hay otro uso importante de la instrucción RETURN, no asociado con subrutinas. (Véase SAS79[6], p. 114, para una descripción).

```
//SYSIN DD *
  DATA A.B.;
    GO TO NEXT; *SALTE SOBRE LA SUBRUTINA;
  *SUBRUTINA PARA COMPUTAR FUNCIONES COMPLICADAS DE
  LA VARIABLE PROMEDIO Y DESVIACION ESTANDAR S.
  EL RESULTADO SE COLOCA EN RESULT;
  COMP:
        Instrucciones para ejecutar las computaciones
  RESULT  =  0;
  RETUNR; * FIN DE SUBRUTINA COMP;
NEXT:
  LENGTH ...:
  INPUT etc.;
  X = AGE; M = MEANAGE; S = SD_AGE;
  LINK COMP;
  AGE_F = RESULT;
  X = WEIGHT; M = MEAN_WT; S = SD_WT;
  LINK COMP;
  WEIGHT_F = RESULT;
```

Fig. 9. Un ejemplo de una Subrutina.

2.9 Procesamiento elemental de Archivos SAS

El propósito básico de la Fase de Entrada de Datos es producir un archivo SAS que pueda ser manipulado con cierta facilidad. En realidad, usualmente no se modifica un archivo SAS que ya ha sido creado.

La "Manipulación" en un paso DATA usualmente involucra una repetición de los siguientes pasos:

a. Leer una obseración de una archivo SAS de entrada.
b. Modificar una observación. (Modificar, agregar y/o suprimir variables; posiblemente suprimir la observación).
c. Escribir la observación modificada al archivo SAS de salida.

Se repite el proceso hasta procesar todas las observaciones del archivo de entrada.

Virtualmente todo el procesamiento de datos SAS es alguna variación de este método. A continuación se discuten la programación del método básico y algunas variaciones.

2.9.1 Procesamiento Básico: DATA y SET

La forma más básica del procesamiento de un archivo SAS está descrita en los párrafos anteriores. Un paso SAS para este procesamiento tiene la forma:

```
DATA outputdsn;
   SET inputdsn;
      { instrucciones para computar nuevas variables,
      { suprimir variables, y/o
      { suprimir observaciones
```

En donde: outputdsn es el nombre del archivo SAS de salida; inputdsn es el nombre del archivo SAS de entrada. Opcionalmente se puede agregrar el parámetro END = varname a la instrucción SET, en donde varname es el nombre de una variable que el supervisor SAS ajustará a 0 por cada observación de entrada excepto la última, y a 1 cuando la última observación es entrada.

Se describe la instrucción detalladamente en las pp. 78-79 del SAS79[6].

Se ejecutan las instrucciones que siguen la instrucción SET para cada observación de entrada.

Concatenando Archivos. Dos o más archivos de entrada pueden concatenarse escribiéndolos, en orden, en la instrucción SET. La forma general es:

```
DATA outputdsn;
   SET inputdsn1 [(IN = var 1)]
       inputdsn2 [(IN - var 2)]
           -
           -
           -
       inputdsnk [(IN = vark)] ;
```

El paréntesis cuadrado indica que el uso del parámetro incluido es opcional.

Al igual que arriba, outputdsn es el nombre del archivo de salida. Los nombres de los archivos de entrada son inputdsn1, inputdsn2, inputdsnk. Si se especifica, var1 es una variable Booleana que el supervisor SAS ajusta a 1 para cada observación de entrada del primer archivo y a 0 para cada observación de los otros archivos.

Con un ejemplo, si el archivo B.MALE contiene datos sobre sujetos masculinos y B.FAMALE contiene datos sobre temas femeninos, se podría crear un juego de datos con datos sobre ambos, de la siguiente manera:

```
DATA B.BOTH;
   SET
      B.FEMALE (IN = IN_FEM)
      B.MALE (IN = IN_MALE);
   IF IN_FEM THEN SEX = 'F'
             ELSE SEX - 'M';
```

2.9.2 Suprimiendo Observaciones: DELETE Subjuegos IF

La instrucción DELETE se usa para suprimir observaciones del archivo de salida. Los verdaderos efectos producidos al ejecutar un DELETE son que el SAS:

a. Cese de procesar la observación actual;
b. No escriba la observación actual al archivo de salida.
c. Inmediatamente comience a procesar la siguiente observación.

Por ejemplo, si el archivo B.BOTH tiene datos masculinos y femeninos y se desea crear un archivo conteniendo solamente datos masculinos, y B.BOTH contiene una variable SEX codificada como 'M' o 'F', el siguiente paso SAS sería apropiado:

```
DATA B.FEMALE;
   SET B.BOTH;
   IF SEX NE 'F' THEN DELETE;
```

La instrucción de subjuego IF puede ser usada para lograr el mismo efecto. El formato general es: "Expresión IF;" en donde expresión es una expresión Booleana. La acción de esta instrucción es exactamente equivalente a:

```
IF expresión THEN;
             ELSE DELETE;
```

(Nótese que la instrucción que sigue a THEN es una instrucción nula). El ejemplo anterior de seleccionar observaciones femeninas podría codificarse:

```
DATA B.FEMALE;
     SET B.BOTH;
     IF SEX = 'F';
```

La ilustración de subjuego de IF es descrito en la p. 104 del SAS79[6]. DELETE e IF funcionan en cualquier paso data.

2.9.3 Seleccionando un Subjuego de Variables: DROP, KEEP

En un paso DATA puede decidirse que el SAS no escriba variables seleccionadas en el archivo de salida. SAS aparta localizaciones en

la memoria para todas las variables definidas para el paso. Para un paso DATA...INPUT, esto incluye todas las variables en las instrucciones INPUT más todas las variables adicionales definidas en las instrucciones dentro del paso. Para otros pasos DATA (DATA...SET, etc.) esto incluye todas las variables en los archivos de entrada más cualesquiera variables adicionales definidas en el paso. Puesto que las localizaciones en la memoria están asignadas a todas las variables definidas, todas las variables definidas están disponibles para computación en todos los puntos en el paso después de que reciben sus valores. (La entrada de las variables de un archivo SAS recibe sus valores 'en' la instrucción SET (o MERGE o UPDATE). La entrada de las variables por la vía de un INPUT reciben sus valores al ejecutarse la instrucción INPUT. Una variable que aparezca a la izquierda de una instrucción de asignación recibe su valor cuando se ejecuta a la instrucción de asignación.

Las instrucciones DROP y KEEP, descritas en la p. 111 del SAS79[6] se usan para restringir el output a un subjuego de las variables. Use la forma "DROP lista de supresión" para especificar que las variables en la lista de supresión no deben de ser escritas en el archivo de salida. Usese la "KEEP lista de retención" para especificar que solamente las variables incluidas en la 'lista de retención' deben escribirse en el archivo de salida. (Se dan ejemplos de DROP y KEEP en el SAS79[6]). No se usa DROP y KEEP en el mismo paso. DROP y KEEP no afectan el almacenamiento en la memoria, sino solamente la lista de variables a ser escrita.

2.9.4 Clasificación, PROC SORT

La clasificación es especialmente fácil en SAS, una vez que los datos están en un archivo SAS. En realidad, en vez de clasificar un archivo se especifica a la vez un archivo SAS de entrada y un archivo SAS de salida. SAS usa programas y procedimientos externos para leer los datos del archivo de entrada, clasifica los datos usando archivos temporales y escribe los datos clasificados, al archivo de salida. El formato general es:

```
PROC SORT DATA = inputdsn;
   OUT = outputdsn;
   BY variables claves;
```

La instrucción PROC SORT tiene opciones adicionales que no se discuten aquí (véase el SAS79[6], pp. 373 y siguientes). Aquí, inputdsn y outputdsn denotan los nombres de los archivos SAS de entrada y salida, respectivamente. Si el outputdsn es igual al inputdsn, el output es colocado en un nuevo archivo SAS. Después de que el paso de clasificación ha terminado con éxito la clasificación y escritura, al archivo de salida se le da el nombre del archivo de entrada y se suprime el archivo de entrada. Si el proceso de clasificación falla debido a falta de espacio en el disco o a la falta de tiempo, por ejemplo, el archivo de salida no será completado, el nombre no será transferido y el efecto será el mismo como si la clasificación nunca hubiese sido intentada.

Excepto en el caso de archivos pequeños y temporales es una práctica de programación pobre el darle el mismo nombre a los archivos de entrada y salida.

Orden de clasificación. El paso PROC SORT tiene que incluir siempre una instrucción BY, el cual lista variables 'claves' que determinan el orden del archivo de salida. El lector debe examinar cuidadosamente el ejemplo PROC SORT del SAS79[6], p. 375. Las variables clave en ese ejemplo son CITY y CHAPTER. CITY es la variable clave mayor y CHAPTER es la variable clave menor. Nótese que las observaciones están ordenadas CHAPTER dentro de CITY.

A menos de que se especifique lo contrario, SAS clasifica los datos a modo que todas las variables en la lista de clasificación BY estén en orden ascendente. Los valores en la primera variable BY estarán en orden estrictamente ascendente; los valores de la segunda variable BY estarán agrupados, de tal forma que todas las observaciones en cada grupo tengan el mismo valor de la primera variable BY. Dentro de un grupo

dado, los valores de la segunda variable BY estarán en estricto orden ascendente. (El ejemplo en la p. 375 del SAS79[6] ilustra el caso de las dos variables BY).

Si se usan más de dos variables BY, los valores de la variable BY (K+1)-st están en orden dentro de cada grupo definido, manteniendo constante los valores de las primeras variables BY K.

Los datos pueden clasificarse en orden descendente en una o más variables BY, poniendo la palabra clave DESCENDING después de dichas variables en la instrucción BY; por ejemplo:

```
BY HEIGHT DESCENDING
   WEIGHT DESCENDING
   AGE;
```

Aquí los valores de HEIGHT del archivo clasificado estarán en orden descendente. Para un grupo de observaciones que tengan el mismo valor de HEIGHT, los valores de WEIGHT estarán en orden descendente. Para un grupo de observaciones que tengan el mismo HEIGHT y WEIGHT, los valores de AGE estarán en orden ascendente. Se le sugiere al lector crear sus propios ejemplos mediante la selección de un archivo de muestra con variables discretas (ej. raza, sexo, etc.), clasificando los datos varias veces, usando diferentes listas BY y diferentes combinaciones de secuencias ascendente y descendente. Usese PROC PRINT para imprimir el archivo después de cada clasificación para observar los resultados.

2.9.5 Las Variables "FIRST" y "LAST" Variables Automáticas

En un paso DATA con un archivo de entrada SAS (e.g., DATA ...SET), si el archivo de entrada está clasificado, es conveniente que el programador pueda determinar cuando el valor de una de las variables de clasificación BY cambia; i.e., determinar:

a. Si una de las variables BY cambió de valor cuando la observación actual se entró (y así la observación actual es la primera con este valor de la variable BY); o

b. Si una de las variables BY va a cambiar cuando la próxima observación sea leída, i.e., si ésta es la última observación con este valor de la variable BY.

SAS provee un mecanismo para tomar estas determinaciones. Para activar este mecanismo se usa una instrucción BY siguiendo a la instrucción SET (o MERGE o UPDATE), de la siguiente manera:

```
DATA outputdsn;
    SET inputdsn;
    BY variables clave;
```

Las variables clave se ponen en lista en la forma que aparecen en la lista "PROC SORT; BY ...;" incluyendo el parámetro DESCENDING, si fue usado.

Considérese el ejemplo:

```
PROC SORT DATA = B.CAFEORIG
          OUT = B.CAFESORT:
    BY PROVINCE CANTON DISTRICT;
DATA TEMP;
    SET B.CAFESORT;
    BY PROVINCE CANTON DISTRICT;
```

En este ejemplo, SAS generará 6 variables Booleanas "automáticas", llamadas:

FIRST.PROVINCE	LAST.PROVINCE
FIRST.CANTON	LAST.CANTON
FIRST.DISTRICT	LAST.DISTRICT

Usualmente se generan dos variables Booleanas automáticas para cada variable en la lista BY. Los valores de estas variables son asignados como sigue:

FIRST.var = 1-si el valor de la var, o cualquier variable que preceda a la var en la lista BY cambió cuando fue entrada la presente observación.
= 0 en caso contrario

LAST.var = 1 si el valor de la var, o cualquier variable que preceda a la var en la lista BY cambiará cuando sea entrada la siguiente observación
= 0 en caso contrario

Como un ejemplo, consideren el próximo cuadro. Los valores de las variables clave aparecen a la izquierda; los valores de las variables automáticas a la derecha. Este archivo fue generado por un segmento de programa como el ejemplo anterior (Cuadro No. 4).

OBS	REGION	CANTON	DISTRICT	FIRST REGION	LAST REGION	FIRST CANTON	LAST CANTON	FIRST DISTRICT	LAST DISTRICT
1	1	1	1			1	0	1	0
2	1	1	1			0	0	0	1
3	1	1	2			0	0	1	1
4	1	1	3			0	1	1	1
5	1	2	1			1	1	1	1
6	1	3	1			1	0	1	0
7	1	3	1			0	1	0	1
8	1	4	1			1	0	1	1
9	1	4	2			0	0	1	0
10	1	4	2			0	1	0	1
11	2	4	2			1	1	1	1

Cuadro No. 4. Valores de variables clave y automáticas.

Nótese que en la observación 5, LAST DISTRIC = 1, aún cuando DISTRIC = 1 tanto en observación 5 como en observación 6. Puesto que DISTRICT está clasificado dentro de CANTON se asume que DISTRIC también cambia cuando CANTON cambia. Así, la observación 5 es la última observación para DISTRICT 1 dentro de CANTON2.

Similarmente, cuando REGION cambia, como de la observación 10 a la observación 11, tanto CANTON como DISTRIC "cambian", aún cuando los valores no parecen cambiar. Esto se debe a que la observación 10 contiene datos sobre el CANTON 4 dentro de la REGION 1, que se supone es diferente al CANTON 4 dentro de la REGION 2, (observación 11).

Nota de programación: Es a menudo deseable saber si solamente hay una observación por cada combinación de valores de las variables clave, i.e., si los valores de los valores clave identifican a una observación como única. En el cuadro anterior, por ejemplo, las variables clave no identifican las observaciones como únicas, ya que las primeras dos observaciones tienen los mismos valores clave.

Se identifica una observación en forma única por las variables BY si y solamente si para la última variable en la lista BY, FIRST.last = LAST.last = 1. En el ejemplo, la última variable en la lista by es DISTRICT; únicamente aquellas observaciones con FIRST.DISTRICT = LAST. DISTRICT = 1 son identificadas en forma única por las variables BY. (Estas son las observaciones 3, 4, 5, 8 y 11).

La instrucción UPDATE, que se discute más abajo, requiere que el archivo "master" tenga las observaciones identificadas en forma única por los valores de las variables BY. Esta técnica puede usarse para verificar esta condición.

2.9.6 Un Ejemplo de Resumen

Se ilustran las ideas y técnicas discutidas arriba, por el segmento de programa en la Fig. 10. En este ejemplo el archivo de entrada es B.CAFESORT, el cual está clasificado BY PROVINCE CANTON DISTRICT IDNUM. Se quiere computar el total de la variable QUANTITY y el promedio de las variables WATER, IN_DIST, OUT_DIST para cada DISTRICT (dentro del CANTON y PROVINCE).

```
DATA B.STATS;
   SET B.CAFESORT; * NOTE= CAFESORT IS SORTED--;
   BY PROVINCE CANTON DISTRICT;
*  "RETAIN" THE VARIABLES USED FOR SUMMING AND COUNTING;
   RETAIN N TOT_QTY SUM_H2O SUM_OUD;
   INITIALIZE THE TOTALS AND N;
   IF FIRST.DISTRICT
      THEN DO;
              N=1
              TOT_QTY=QUANTITY;
              SUM_H2O=WATER
              SUM_IND=IN_DIST;
              SUM_OUT=OUT_DIST;
           END;
      ELSE DO;
              * FOR SUBSEQUENT OBSERVATIONS
                INCREMENT TOTALS;
              N=N+1
              TOT_QTY = TOT_QTY + QUANTITY;
              SUM_H2O = SUM_H2O + WATER;
              SUM_IND = SUM_IND + IND_DIST;
              SUM_OUT = SUM_OUT + OUT_DIST;
           END;
* AT THE LAST OBSERVATION FOR THE DISTRICT,
  COMPUTE MEANS AND OUTPUT AND OBSERVATION;
  IF LAST. DISTRICT
      THEN DO;
              MEAN_H2O = SUM_H2O/N;
              MEAN_IND = SUM_IND/N;
              MEAN_OUT = MEAN_OUT/N;
              OUTPUT;
           END;
      ELSE DELETE; * WE OUTPUT AN OBSERVATION ONLY
                     AT THE END OF THE DISTRICT;
  KEEP PROVINCE CANTON    DISTRICT  N
       TOT_QTY  MEAN_H2O  MEAN_IND  MEAN_OUT;
```

Fig. 10. Una ilustración del uso de las variables "FIRST", y "LAST." con archivo SAS clasificado.

El archivo de salida B.STATS, debe contener las variables PROVINCE, CANTON, DISTRICT, TOT_QTY, (total de QUANTITY), MEAN_H2O, MEAN_IND, MEAN_OUT (medias de WATER, IN_DIST, y OUT_DIST, respectivamente), y N, el número de observaciones usadas para computar la suma y las medias. Se asume que no hay valores omitidos.

Ejercicio. Inventen valores de datos para las variables WATER, QUANTITY, IN_DIST para cada una de las observaciones mostradas en el último Cuadro en la Sección 2.9.5. A mano (no en la computadora!), tracen la ejecución del programa en la Fig. 10 y 'produzcan' el archivo de salida. Verifiquen que comprenden exactamente como funcionan las varias características del programa.

Nota. Hay maneras más eficientes para realizar la tarea de la Fig. 10 en SAS, incluyendo PROC SUMMARY y PROC MEANS. El propósito del ejemplo es ilustrar las características discutidas en este capítulo y las técnicas de programación relacionadas.

2.10 Procesamiento Avanzado del Archivo SAS, MERGE, UPDATE

El lector está ahora lo suficientemente preparado para estudiar el capítulo 9 del SAS79[6], el cual discute en detalle las instrucciones SET, MERGE y UPDATE. La descripción del capítulo 9 del SAS79[6] es adecuada para los propósitos de esta sección.

2.11 PROCs para Manipulación de Archivos

SAS provee varios PROCs para asistir en la manipulación de archivos OS y archivos SAS. El alumno debe ahora estudiar las siguientes secciones del SAS79[6].

Capítulo 14, "Estudio Sumario: El paso PROC", p. 119

PROC CONTENTS	p. 163
PROC COPY	p. 171
PROC DATASETS	p. 179
PROC DELETE	p. 181
PROC MEANS	p. 303
PROC PRINT	p. 353
PROC RELEASE	p. 365
PROC SUMMARY	p. 397
PROC TAPECOPY	p. 419
PROC UNIVARIATE	p. 427

Los procedimientos estadísticos en la lista de arriba, MEANS, SUMMARY y UNIVARIATE son útiles para revisar un archivo buscando valores inapropiados y otros. Además, MEANS y SUMMARY son útiles para resumir un archivo, por ejemplo un archivo es entrado y se computan estadísticas resumidas que son entonces enviadas a otro archivo de salida. La salida del archivo está entonces disponible para procesamiento posterior.

CAPITULO 3

UN MODELO GENERAL DE UN SISTEMA DE MANEJO DE DATOS DE INVESTIGACION

3.1 Introducción: Objetivos

El propósito de esta sección es presentar un modelo general de un sistema MDI. La palabra 'modelo' no necesariamente significa que el sistema es ideal y que podría reproducirse exactamente para cada problema que se presente. Más bien, el sistema es modelo en el sentido de que tiene la mayoría de las características que se considerarían para inclusión en un sistema que se esté diseñando. Para un problema particular hay que seleccionar lo apropiado del sistema y adaptarlo a las necesidades del problema dado.

Las siguientes secciones contienen un diagrama de flujo del sistema modelo y la descripción de todos los subsistemas y componentes. Se discute la implementación del sistema en SAS en aquellas secciones en donde los métodos de implementación pueden ser menos obvios.

3.2 Diagrama de Flujo y los Componentes Principales del Sistema Modelo

Se presenta un diagrama de flujo del sistema modelo en las Figuras 11 y 12. El sistema tiene cinco componentes principales:

- El Procesamiento Manual Preliminar
- El Sistema Principal para el Manejo de Datos (detección de errores, mantenimiento de archivo)
- El Subsistema de Inventario
- El Subsistema de Corrección de Errores
- El Subsistema de Control de Calidad (no aparece en el diagrama de flujo)

En el diagrama de flujo se indica el Sistema de Manejo de Datos (DMS) principal por medio de una casilla hecha con líneas entrecortadas (Fig. 11). El Procesamiento Manual Preliminar consiste en aquellos pasos situados arriba de la casilla del DMS en dicha figura. El Subsistema de Inventario está a la izquierda de la casilla del DMS principal, y el Subsistema de Corrección de

Errores aparece en el diagrama de la Fig. 12. A continuación se describen detalladamente estos componentes.

3.3 Procesamiento del Archivo Plano Distribuido

Helms[4] discute la estrategia del "Archivo Plano Distribuido" para problemas del RDM con múltiple corriente de datos. Sus puntos básicos son:

Muchos proyectos involucran la colección de varios tipos diferentes de datos a través de un período si los diferentes tipos de datos (formas de datos) se recogen todos en la misma fecha calendario, entonces los datos son fácilmente encadenados por período y todos los datos con la misma fecha calendario constituyen una corriente de datos. Sin embargo, en muchos casos se recogen diferentes tipos de datos en diferentes fechas calendario. Por ejemplo, un estudio puede tener un tipo de datos colectado mensualmente, otro tipo trimestralmente, otro anualmente, y todavía más, aún otro tipo que se recoge cuando un evento específico ocurre, i.e., esencialmente al azar. En tal caso cada tipo de datos constituye una corriente de datos y el proyecto total tiene varias corrientes de datos. Para la fase de procesamiento de datos de producción, Helms[4] ha mostrado que el preparar un sistema RDM para cada corriente es a menudo una buena estrategia, lo que se llama estrategia de los "Archivos Planos Distribuidos" (DFF).

Si se usa la estrategia DFF, en lugar de diseñar e implementar un sistema grande y complicado, se obtiene un sistema mucho más simple para cada corriente de datos.

El sistema modelo que se describe en las siguientes secciones no asume ni la estrategia DFF ni lo inverso, o sea la "estrategia sistema monolítico". Para usar la estrategia DFF simplemente se implementan sistemas separados para cada corriente de datos. La estrategia del sistema monolítico puede ser implementada al diseñar un componente MDI capaz de manejar todas las corrientes simultáneamente.

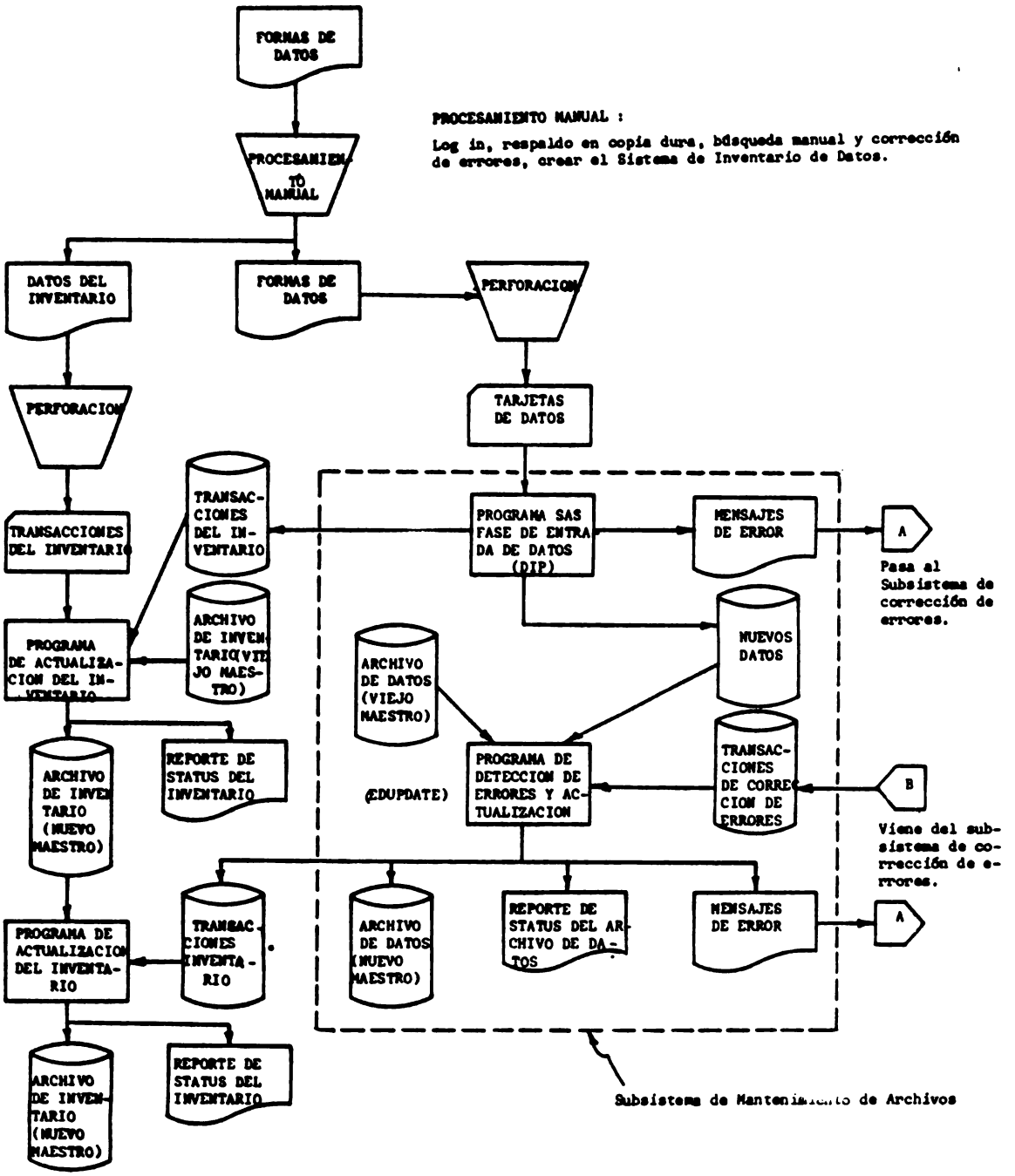


Fig. 11. Diagrama de Flujo de un "modelo" de Sistemas de Manejo de Datos.

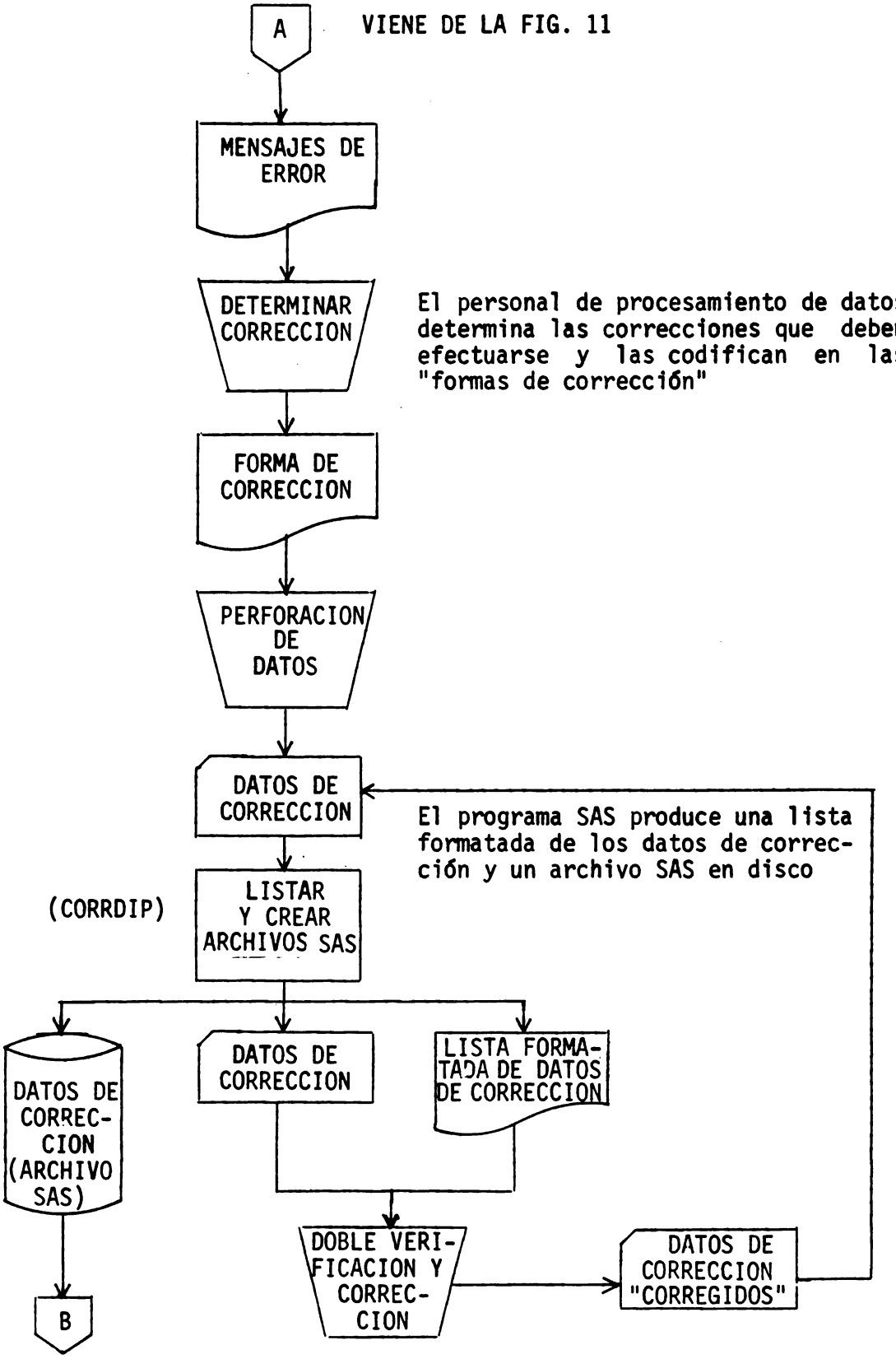


Fig. 12. Subsistema de Corrección de Errores de un "Modelo" de Sistema de Manejo de Datos

3.4 Procesamiento Manual Preliminar

Las tareas involucradas en el procesamiento manual preliminar dependen mucho de factores locales como la fuente de datos (interna o externa a la organización), si se completan las formas de datos localmente o en otro lugar, la naturaleza del estudio (longitudinal o sección transversal) y otros.

3.4.1 Log in

Las formas se reciben, comúnmente, en un paquete que puede contener varios tipos de formas de diferentes corrientes de datos. Uno de los primeros pasos manuales es el de log in las formas, estableciendo un registro escrito de la existencia y arribo de cada forma. Típicamente este paso de log in crea datos para el Subsistema de Inventario, el cual mantiene información sobre la presencia, localización y condición de procesamiento de los registros de datos. Los datos de inventario se codifican inmediatamente y los datos de proyectos se someten a procesamiento adicional antes de ser codificados.

3.4.2 Respaldo en Copia Dura (Hard copy)

Si los datos tienen algún valor sustancial, el próximo paso es sacar un 'copia dura de respaldo' de las formas de los datos. Para proyectos muy pequeños la copia puede ser por medio de xerografía u otro fotocopiado. Para estudios más extensos el microfilm es menos costoso y requiere menos espacio de almacenaje, aunque hay una inversión inicial adicional por el equipo. Otra ventaja del microfilm es que las copias adicionales son muy baratas; y es práctico obtener y guardar una copia adicional en un lugar seguro y remoto. En el caso de un incendio en la instalación de procesamiento, la copia de respaldo en el lugar remoto puede ser la única copia restante de los datos.

3.4.3 Búsqueda de Errores y Corrección Manual

Antes de codificar, las formas de datos son revisadas cuidadosamente por una persona familiarizada con el proyecto y sus datos. La

corrección de errores en esta etapa es menos costosa y menos problemática que hacerle correcciones a un archivo de la computadora: el revisar cuidadosamente verifica, en cada forma, su legibilidad, el cumplimiento con instrucciones, los datos obviamente incorrectos o inconsistentes, los datos omitidos, y otros elementos.

La verificación más importante en la búsqueda manual de errores es la de errores en el campo clave. Un campo clave es un campo de datos que luego se usará para clasificación y/o identificación de un registro en particular, un asunto en particular. Los campos clave son a menudo llamados campos de identificación y frecuentemente aparecen en las instrucciones BY en los programas SAS. A los campos no clave se les dice 'campos de datos'.

Los errores de campo clave son mucho más difíciles de corregir que los errores en el campo de datos una vez que los datos ya están en un archivo de la computadora. El procedimiento para la corrección de un error en el campo de datos es mediante la entrada de un valor nuevo y correcto y la actualización de la observación (descrito más adelante). Para 'corregir' un campo clave, es necesario hacer una observación enteramente nueva (perforar nuevas tarjetas), agregar la nueva observación al archivo y suprimir la vieja observación con los valores clave erróneos.

El revisor de datos manual debe examinar cuidadosamente los valores del campo clave y determinar su corrección absoluta, pues no es suficiente, sólo el simplemente determinar que los valores del campo clave están en una extensión válida.

Cualquier error detectado en una búsqueda manual de errores es corregido antes de que los datos sean perforados. Usualmente el revisor de datos tiene que ponerse en contacto con las personas que los escribieron en las formas, para poder determinar las correcciones.

Nuevos errores pueden ser accidentalmente introducidos cuando se hacen correcciones. Todos los cambios de datos hechos por el revisor son verificados por un supervisor antes de que las correcciones sean introducidas en el sistema.

Después de la búsqueda manual de errores y correcciones (si son necesarias) los datos son enviados para digitación. Nótese que los datos de inventario creados por el proceso log-in fueron enviados a ser digitados inmediatamente después de log-in.

3.4.4 Digitación y Verificación

Los datos son digitados (perforación, teclado-a-discos o equivalente) y verificados. Las personas que procesan datos no verificados pierden tiempo y dinero ---numéros al azar, el producto de codificación no verificada, pueden ser creados mucho más rápidamente y mucho más barato por la computadora. Si se desea procesar tales números al azar, ¿para qué molestarse con formas de datos y digitación?

La salida del paso de digitación-y-verificación se representa en el diagrama de flujo (Fig. 11) como una tarjeta de datos. Esto es solamente simbólico; si se usa teclado-a-disco o teclado-a-cinta puede sustituirse por un símbolo apropiado.

3.5 Enfasis en Seguridad y Detección y Corrección de Errores

El lector rápidamente observará que una de las diferencias entre el manejo de datos por un profesional o un aficionado es la preocupación del profesional por la calidad y fidelidad de los datos. Estas preocupaciones son manifiestas en la extensión de las facilidades de respaldo de los datos y detección y corrección de errores en el sistema.

En muchos proyectos de investigación hay un punto después que los datos han sido colectados y antes de que análisis sustanciales hayan sido completados, en el cual casi toda la inversión en el estudio --la inversión financiera, la inversión significativa del tiempo de los científicos participantes , y la inversión en potencial humano-- está concentrada en los archivos de datos en unas pocas cintas o discos. El destruir estos archivos es destruir casi toda la inversión del estudio.

Los archivos de datos pueden ser destruídos por desastres físicos, tales como fuego o inundación o por mal funcionamiento de la computadora ('crash' del disco, datos escritos sobre otros datos, etc.). Los archivos de respaldo son

la protección apropiada contra este tipo de desastre. Este tema será discutido en la sección sobre Seguridad de los Datos.

También puede efectivamente destruirse la inversión del proyecto introduciendo errores al azar (o sistemáticos) dentro de los datos y no removiéndolos. Los errores son introducidos en cada etapa de transcripción humana o procesamiento. (Las computadoras también introducen errores, pero eso es otro tema!). Los errores se introducen cuando 'se recogen' los datos (como errores de respondente en las entrevistas), cuando las fórmulas se llenan y cuando se hacen "correcciones". El asunto no es saber si acaso son introducidos errores en cada una de estas fases, sino ¿con cuánta frecuencia? La cuestión es: ¿cuál es el promedio de error en cada transcripción humana o proceso?

Si el promedio acumulado de errores no corregidos es muy alto, se destruye la inversión del proyecto en el estudio casi tan seguro como si se hubieran quemado las copias de los datos. La destrucción por errores es aún más insidiosa que por fuego, ya que de esta última se da cuenta inmediatamente. Cuando un archivo es destruído por errores sus resultados podrán ser publicados en la literatura científica y las conclusiones erróneas pueden ser descubiertas por varios años.

Los errores de datos son importantes. El lector notará que un componente sustancial del sistema está dedicado a la detección y corrección de errores y a la estimación del promedio de error (subsistema de control de calidad).

3.6 Detección de Error

Los objetivos principales del Sistema de Manejo de Datos (DMS) son:

- Detectar errores y proveer su corrección.
- Mantener (crear, actualizar) el archivo maestro de datos.
- Producir informes para los gerentes de proyectos sobre la condición del procesamiento de los datos.
- Proveer la seguridad de los datos por medio del respaldo sistemático de los archivos.

Un sistema puede ser aplicado en uno o varios programas de computadora. En una aplicación SAS tal 'programa' usualmente incluye varios pasos SAS.

La detección de errores se realiza en dos partes en el diagrama de flujo de la Fig. 11 en el Programa de la Fase de Entrada de Datos y en el Programa de Actualización y Detección de Errores.

El propósito principal de la detección de errores en el Programa de Fase de Entrada de Datos es identificar errores estructurales, tales como valores inválidos en el campo clave y 'tarjetas omitidas'. El principal esfuerzo de detección de errores computarizado del sistema se concentra en el componente de detección de errores del Programa de Detección de Error y Actualización. Este componente somete los datos a las pruebas prácticas más rigurosas dentro de la estructura del proyecto dado.

El resto de esta sección describe las técnicas de detección de errores, de las cuales dos tipos básicos de pruebas de errores: pruebas de campo y pruebas de consistencia. Una prueba de campo se basa solamente en el conocimiento de valores permisibles para el campo de datos (variable). Se usan tres tipos de pruebas de campo comúnmente: (1) valores válidos, (2) alcances válidos y (3) definición del tipo de campo.

3.6.1 Pruebas de Campo de Valores Válidos

Se usa una prueba de campo de valores válidos cuando el juego completo de valores válidos para una variable (campo) es conocido y tiene pocos elementos. Considérese, por ejemplo, el item de una forma de datos:

```
17.  Sexo del sujeto:
     1.  Masculino..... M
     2.  Femenino...... F
```

Se dan solamente dos valores válidos para la variable SEXO en este item. La prueba de campo de valor válido determinaría si el valor de los datos es un valor válido, e.g.,

```
IF NOT (SEX = 'M' or SEX = 'F')
   THEN toma acción por error;
```

Todavía no se ha discutido 'tomar acción por error', pero por lo menos hay un ejemplo. Otra versión de esta instrucción usa una variable Booleana:

```
ERR1 = NOT (SEX = 'M'
       OR SEX = 'F');
```

Este tipo de instrucción puede ser 'mecanizado' (para facilidad de perforación del programa) para variables con más de dos valores válidos. Supóngase que ITEM tiene los valores válidos 1, 2, 3, 4, y 9; la siguiente instrucción contiene la parte esencial de la prueba de valor válido:

```
ERR2 = NOT (ITEM = 1
       OR ITEM = 2
       OR ITEM = 3
       OR ITEM = 4
       OR ITEM = 9);
```

La prueba del valor válido tiende a producir programas SAS largos por razones obvias. El hacer programas en formatos apropiados mejora la legibilidad, reduce errores y acorta el tiempo de búsqueda de problemas y pruebas.

3.6.2 Pruebas de campo de alcanes válidos

Las variables numéricas que se conocen y que corresponden a un alcance dado, y las cuales tienen más valores válidos de los que razonablemente pueden ser comprobados por una prueba de valores válidos, pueden verificarse por una prueba de alcance válido.

Como ejemplo considérese una fecha que se sabe cae entre 20OCT73 y 05JAN75 y para lo cual el año, mes y día se han entrado separadamente dentro de las variables números YEAR, MONTH y DAY respectivamente.

Si una fecha está sujeta a error es buena práctica de programación el entrar los componentes por separado y realizar la conversión a una fecha variable internamente, en donde el programa puede controlar los efectos de errores. Los errores en la conversión del formato de fecha SAS resultan en un valor omitido para la fecha --una situación que está más allá del control del programador. El segmento de programa en la Fig. 13 realiza pruebas de extensión válida sobre MONTH, DAY y YEAR; si los valores individuales son válidos, convierte estos valores a una fecha variable y verifica su alcance. Nótese que el programa deja una revisión detallada del número de días en el año a la función SAS MDY (ver SAS79[6], p. 41). MDY devolverá un valor omitido si la combinación de mes y día es inválida.

3.6.3 Pruebas de Consistencia

Una prueba de campo compara el valor de una variable con los valores permitidos para esa variable, sin considerar otros factores. Una prueba de consistencia usa comparaciones de los valores de dos o más variables. El ejemplo de las fechas es un buen ejemplo. Dado un MONTH, DAY y YEAR, como en la Fig. 13 una prueba de campo para DAY tiene que usar sólo el alcance válido 1 DAY 31. Una prueba de consistencia inválidas, tales como Febrero 30. Así, DAY = 30 pasará la prueba de campo de valor válido, pero la combinación MONTH = 2 y DAY = 30 fallaría la prueba de consistencia. (El ejemplo clásico en datos médicos es una comparación de SEX versus PREGNANT, para verificar si hay varones embarazados).

La información para una prueba de consistencia puede conocer a priori con base en ciertas reglas, como en el ejemplo de la fecha, o de resultados publicados previamente. Además, las pruebas de consistencia pueden generarse de datos. Considérense dos variables números, X y Y, las cuales tienen un alto coeficiente de correlación linear. Se puede realizar un análisis de regresión para obtener coeficientes de regresión

```
          DATE_ERR = 0;
          IF YEAR < 73 OR YEAR > 75
             THEN DO;
                  PUT / 'ERROR IN YEAR';
                  DATE_ERR = 1;
             END;
          IF MONTH < 1 OR MONTH > 12
             THEN DO;
                  PUT / 'ERROR IN MONTH';
                  DATE_ERR = 1;
             END;
          *  PRUEBA APROXIMADA DE FECHAS VALIDAS.  SAS
          *  EJECUTA LAS PRUEBAS EXACTAS EN LA FUNCION MDY;
          IF DAY < 1 OR DAY > 31 THEN GO TO ERR_D;
          MYDATE = MDY(MONTH, DAY, YEAR);
          IF MYDATE NE . THEN GO TO D_OK:
ERR_D:    PUT / 'ERROR IN DAY';
          DATE_ERR = 1;
          GO TO BOTTOM;
D_OK:     *  PRUEBA DE ALCANCE ACTUAL DE FECHAS;
          IF MYDATE < MDY(10, 30, 73) OR
             MYDATE > MDY(1, 5, 75)
             THEN DO;
                  PUT / 'MYDATE OUT OF RANGE 30 OCT 73'
                        'Ø TO 05 JAN 75';
                  ERR_DATE = 1;
             END;
BOTTON:   IF ERR_DATE THEN LIST;
```

Fig. 13. Un segmento de programa que ilustra la prueba de Alcances Válidos para Date (YEAR, MONTH, DAY) por Validez y que su alcance caiga entre 30 OCT 73 y 05 JAN 75.

y bandas de confianza de 99.9% para una futura observación de Y, dado un valor de X.

La prueba consiste en determinar para un valor dado de X, si Y se encuentra o no dentro del intervalo de confianza de 99.9%. Si es así la prueba pasa; de lo contrario la prueba falla. Desde luego, se puede usar coeficientes de confianza diferentes de 99.9%.

Los datos para determinar las pruebas de consistencia pueden provenir de análisis preliminares de datos que se están procesando, de datos previos o de resultados publicados previamente para datos similares.

3.6.4 Errores y Valores Improbables

El proceso de detección de errores es poco exacto. Por ejemplo, una fecha que debería ser (en formato yymmdd) 800105 y está como 800205, posiblemente no se detecte con ninguna prueba. En contraste, un valor raro tal como HEIGHT = 225 cm para una hembra humana, debe fallar una prueba de valor válido o de consistencia, aún si el valor es correcto.

Los objetivos de las estrategias y procedimientos para la detección de errores son:

- El detectar tantos errores como sea posible.
- Consistente con un costo de procesamiento razonable.

En una prueba de alcance válido para HEIGHT humano, por ejemplo, al hacer el alcance válido muy corto, virtualmente todos los errores muy grandes (diferencia entre la altura verdadera y la altura dada) serán reportados, pero un gran número de valores correctos serán reportados como posibles errores. Es costoso el verificar un mensaje de error y actualizar el archivo para que refleje los resultados de la revisión.

Una razonable detección de errores necesariamente involucra intercambio en el costo de no detectar errores que existen en los datos y el costo de procesar mensajes de error para datos correctos.

Una forma apropiada es definir las pruebas que detectan una razonable proporción de valores improbables, sean correctos o no. Un juicio profesional es requerido para determinar la proporción de valores correctos que se reportarán como "errores" (valores improbables).

3.6.5 Información Redundante

Toda verificación de errores se basa en información total o parcialmente redundante. En el ejemplo de la prueba de valor válido de la variable SEX, los dos elementos de información redundante son:

1. El valor de SEX
2. Los valores válidos de SEX

Para otro ejemplo, puesto que la altura y el peso humano son altamente correlacionados, estas variables son parcialmente redundantes y pueden usarse para una prueba de consistencia. Las variables que no están relacionadas del todo, no pueden ser usadas para una prueba de consistencia significativa. Cuanto más redundante la información, i.e., cuanto más cercana la relación entre dos (o más) variables, tanto mejor será la prueba resultante.

El principio de basar las pruebas en información redundante es importante en el diseño de formas e instrumentos de colección de datos. Datos completamente redundantes serán colectados de variables que son crucialmente importantes para un estudio. Los químicos, por ejemplo, rutinariamente hacen tres (o más) pruebas de un ensayo importante; si uno de los valores difiere marcadamente de los demás, se toma una acción correctiva. Cuando se hacen tales determinaciones múltiples, deben ser incluídas en los datos.

Se usa la redundancia parcial solamente para variables menos importantes. La redundancia puede provenir del conocimiento sobre la variable (como en la prueba de valor válido) o del conocimiento de la relación apropiada entre dos o más valores, como en una prueba de consistencia.

Al diseñar formas de datos y un sistema de manejo de datos, tiene que determinarse la importancia relativa de cada variable a ser colectada. Información totalmente redundante tiene que ser colectada sobre variables de importancia crucial. Información parcialmente redundante se colecta sobre variables de menor importancia. Las variables que no son lo suficientemente importantes para justificar alguna redundancia y revisión cuidadosa deben omitirse del estudio.

3.7 Qué hacer con datos "sucios"

¿Qué debe hacer el sistema con las observaciones que contiene errores detectados o "valores improbables"? Hay dos estrategias básicas.

3.7.1 Archivo limpio --Archivo sucio--

Una estrategia para tratar el problema es el producir dos archivos "maestros". Todas las observaciones que no tienen erroes detectados se colocan en un archivo "limpio". Todas las observaciones que contienen errores detectados, o valores improbables, se colocan en un archivo "sucio". Las correcciones de errores se hacen a las observaciones en el archivo sucio. Cuando una observación ha sido "purificada" (todos los errores detectados corregidos y todos los valores correctos pero improbables verificados), se pasa al archivo limpio efectuando una actualización.

La estrategia del archivo limpio --archivo sucio tiene la ventaja de que los análisis preliminares pueden realizarse con el archivo limpio sin ninguna precaución en particular acerca de datos erróneos.

En algunos proyectos, sin embargo, la corrección de errores puede requerir tiempo sustancial, especialmente si el volumen de los datos es grande y el centro de colección de datos se encuentra alejado del centro de procesamiento de datos. En tal caso, los datos se acumulan en el archivo sucio y el movimiento al archivo limpio es muy lento. Además, si una observación contiene muchas variables, es muy probable de que al menos una variable fallará en una prueba, ya sea que el valor es correcto pero improbable o que esté en error. Aquí, de nuevo, las observaciones tienden a acumularse en el archivo sucio y se mueven lentamente al archivo limpio. El fracaso de variables relativamente sin importancia en pasar las pruebas pueden mantener a variables correctas e importantes fuera del archivo limpio por meses.

3.7.2 Bytes de Status

Una estrategia alterna, aparentemente usada primero para datos de investigación por Helms[2] utiliza un solo archivo maestro de datos, el cual contiene todos los registros de datos disponibles. Cada

registro ("observación" en SAS) contiene un "indicador de status" para cada variable de datos el indicador contiene información acerca de la calidad de la variable asociada (por conveniencia, los sistemas que utilizan esta técnica usualmente usan un byte para almacenar el valor de cada indicador de status, y el término byte de status se ha popularizado; esquemas más compactos son posibles porque el número de los status posibles es pequeño).

El programa de detección de errores y actualización ajusta los valores de los bytes de status. Al comienzo de la comprobación de error de una nueva observación todos los bytes de status son indizados a valores indicando 'no error detectado' (ver Fig. 14 para un típico juego de códigos de byte de status). Si una variable falla las pruebas de valor válido, alcance válido o de consistencia, el byte de status se ajusta de manera correspondiente (y los mensajes de error se imprimen).

Los bytes de status pueden ser reajustados como resultado de la corrección de errores y del proceso de actualización, descrito en una subsiguiente sección.

Con la técnica del byte de status todos los datos están disponibles para un análisis preliminar. El analista puede determinar, sobre una base de variable a variable, cuán 'limpios' deben ser los datos para entrar al análisis. Si hay errores en variables no involucradas en el análisis, los datos 'limpios' en la observación pueden aún ser usados. En contraste, bajo la estrategia archivo limpio --archivo sucio, solamente los registros que están completamente limpios pueden ser usados.

Ninguna de las estrategias es siempre superior; la estrategia a escogerse dependerá de las circunstancias del proyecto específico.

3.8 Mantenimiento de Archivo

Aunque esencialmente distintas, la función de mantenimiento de archivo y la función de detención de errores, son realizados en el mismo programa de Detección de Error y Mantenimiento. Para el mantenimiento de archivo uno

Código	Interpretación
'b' (blanco)	No error detectado para este valor.
'c'	Este valor falló una o más pruebas pero un ser humano subsecuentemente determinó que era correcto. El sistema no cambiará este código.
'C'	Este valor ha fallado una prueba de consistencia.
'M'	El valor de los datos está omitido.
'V'	Este valor ha fallado una prueba de valor válido o de alcance válido.

Nota: En la secuencia de comparación de IBM 370 'b'<'c'<'C'<'M'<'V'. Este ordenamiento de los códigos refleja creciente severidad de 'error' o decreciente calidad estimada del valor.

Fig. 14. Un Típico Juego de Códigos de Byte de Status

podría escoger ya sea acceso directo o secuencial a los archivos maestros. SAS Apoya sólamente el procesamiento secuencial, el cual es discutido aquí.

El proceso básico de mantenimiento de archivo se fundamenta en las instrucciones SAS DATA y UPDATE. El procedimiento involucra el copiado de datos de un archivo "maestro viejo" (la versión más al día de los datos disponibles para entrada) a un archivo "maestro nuevo". Durante el proceso de copiado se hacen modificaciones, como es indicado por 'transacciones de actualización' (observaciones) de un 'archivo de transacciones'.

Los tres archivos contienen las mismas variables clave (BY) y todos están ordenados en la misma secuencia. El procedimiento de actualización se describe en algún detalle en el capítulo 9 del SAS79[6], especialmente en las pp. 85-88.

3.9 El Programa de Detección de Error y Actualización "EDUPDATE"

El programa clave en el sistema es el de Detección de Error y Actualización (EDUPDATE). Este programa puede definirse en términos de sus entradas, sus salidas y su procesamiento, los cuales se describen a continuación.

3.9.1 Entradas

Como se ven en la Fig. 11, las entradas al EDUPDATE son los archivos NEW DATA ("NEWDATA"), ERROR CORRECT TRANSACTION ("ERRCORR") y OLD MASTER ("OLDMASTR").

Como un paso SAS DATA--UPDATE puede procesar solamente un archivo maestro viejo y un archivo de transacción, un paso SAS preliminar es ejecutado similar a:

```
DATA TEMP;
      SET B. NEWDATA
          B. ERRCORR;
      BY variables clave;
```

El archivo temporal TEMP es entonces usado como entrada para el paso principal de EDUPDATE. Sin embargo, es más conveniente describir NEWDATA y ERRCORR separadamente.

NEWDATA. Este archivo contiene nuevos datos, ordenados por las variables clave. El programa Fase de Entrada de Datos ("DIP") ya ha verificado que:

1. Las variables clave pasan pruebas de campo apropiadas.
2. Cada registro del archivo es identificado en forma única por sus variables clave.

El programa DIP también agrega una variable (a las variables clave y de datos): INPUTDAY, con la fecha y la hora en que el programa DIP fue ejecutado.

ERRCORR. El formato de este archivo SAS dependerá en gran medida de los detalles del Subsistema de Corrección de Error, descrito en otra sección separada. Si se usa la forma de corrección de error por digitación "turn around", descrita en esa sección, el archivo ERRCORR típicamente será como sigue.

Cada observación contiene las variables clave, de datos, y de byte de status con los mismos tipos y longitudes como en el OLDMASTR. La observación también contiene las siguientes variables adicionales: CORRTIME, una variable SAS fecha-tiempo, conteniendo la fecha y la hora en que el programa DIP de corrección de error fue ejecutado, FLAG, una variable carácter de un byte, que es entrada de la forma de corrección de error, codificada como sigue:

1. 'C' indica que esta observación contiene una corrección de valor de datos para reemplazar el valor de una variable en el OLDMASTR.
2. 'D' indica que la observación en el OLDMASTR con valores clave igualando este registro debe eliminarse.
3. 'b' indica que ésta es una observación de datos nuevos. Las observaciones de datos nuevos entran vía la corriente de corrección cuando, por ejemplo, un error es descubierto en los valores clave. El registro viejo es eliminado y se entra una nueva observación, con valores clave correctos.

Típicamente una observación ERRCORR es generada para cada corrección de datos. Por ejemplo, si tres campos se están corrigiendo para una observación OLDMASTR, ERRCORR contendrá una observación para cada uno. Así, en una observación ERRCORR la mayoría de las variables tendrán valores omitidos. El Subsistema de Corrección de Error puede combinar todas las observaciones ERRCORR para una observación OLDMASTR dentro de una sola observación.

ERRCORR es ordenado por las variables clave y FLAG.

Nótese que CORRDIP, el programa Fase de Entrada de Datos de Corrección almacena tanto los valores de datos como los bytes de status en ERRCORR. Los nombres de las variables bytes de status de ERRCORR

son diferentes de las variables byte de status OLDMASTR de manera que los bytes de status del OLDMASTR tienen que ser explícitamente actualizados por EDUPDATE. (De lo contrario, bytes de status blancos del ERRCORR escribirían encima de bytes de status no blancos del OLDMASTR durante la fase de actualización SAS).

OLDMASTR es la versión más recientemente creada del archivo de datos maestro. Contiene todas las variables clave, de datos, status, y fecha-hora arriba descritos, más:

1. UPDATED, una variable fecha-hora ajustada a la fecha-hora en que una observación es agregada al OLDMASTR o modificada por EDUPDATE.
2. FLAG, no está incluída en OLDMASTR o NEWMASTR.
3. CLEAN, una variable Booleana con valor 1 si la observación es 'limpia', o lo contrario. 'Limpia' significa que todas las variables tienen status < = 'c'.
4. CHECKED, una variable fecha-hora indicando la fecha-hora en que la detección de error fue realizada.

Desde luego, OLDMASTR está clasificado por las variables clave.

3.9.2 Salidas

EDUPDATE crea dos archivos de salida SAS, NEWMASTR y INVTX, y dos o más archivos de impresión, incluyendo MESSAGES y ERRORS.

NEWMASTER contiene las mismas variables que OLDMASTR y está ordenado en la misma secuencia.

INVTX, el archivo de Transacción del Archivo de Inventario, contiene una observación por cada:

1. Nueva observación agregada al archivo maestro.
2. Observación del archivo maestro modificada por EDUPDATE.

INVTX contiene las variables clave, más las variables UPDATED, CLEAN y CHECKED. Se describe el uso de este archivo en la sección del Subsistema de Inventario.

El archivo de impresión ERRORS contiene mensajes de datos-error para "errores' detectados por EDUPDATE. Esta impresión es descrita en la sección del Subsistema de Corrección de Error.

MESSAGES. Este archivo de impresión contiene un informe del proceso de actualización, incluyendo:

El número total de observaciones en OLDMASTR
El número total de observaciones suprimidas
El número total de nuevas observaciones agregadas
El número total de observaciones en NEWMASTR

(Estos totales permiten una determinación de observaciones perdidas o sí erróneamente se han agregado entre o durante corridas de actualización). MESSAGES también contiene el número total de observaciones 'limpias', el número total de correcciones y otros datos estadísticos interesantes.

DELETES. Este archivo de impresión contiene una lista formatada de todas las observaciones suprimidas durante la corrida de actualización.

PROBLEMS. Este archivo de impresión contiene una lista formatada de observaciones involucradas en aparentes transacciones erróneas, como por ejemplo:

1. Cuando una observación de corrección o supresión aparece en ERRCORR sin ninguna observación comparable en OLDMASTR.
2. Cuando una observación de datos nuevos actualiza una observación que previamente estaba en OLDMASTR.
3. Solamente observaciones de corrección o supresión deben actualizar una observación en OLDMASTR.

Otros problemas de actualización también son reportados en este archivo.

3.9.3 Procesamiento de EDUPDATE

Como se notó en la sección de entradas de EDUPDATE, éste contiene un paso de preprocesamiento para combinar NEWDATA y ERRCORR dentro de un archivo TEMP.

El segundo paso SAS es típicamente el paso de actualización, con una estructura básica indicada por lo siguiente (Fig. 15):

```
DATA B.MASTER14
     INVENTOR TX27 (KEEP = KEY UPDATED CLEAN CHECKED);
     UPDATE  B MASTER 13 (IN = IN_OLDM)
             TEMP        (IN = IN_T);
     *MASTER14 ES NEWMASTR;
     *MASTER13 ES OLDMASTR;
     *TX27 CONTIENE TRANSACCIONES DEL ARCHIVO DE INVENTARIO
      DE SALIDA;
     *TEMP CONTIENE NEWDATA Y ERRCORR;
```

Esta es una sección de código para la revisión y manejo de problemas de procesamiento. Ver descripción del archivo PROBLEMS.

Este bloque representa una sección de código que procesa una observación DELETE.

Este bloque representa una sección de código que procesa actualizaciones de corrección, incluyendo el mover nueva información de byte de status, dentro de variables de byte de status de MASTER.

Este bloque representa una sección de código que realiza pruebas de campo sobre campos nuevos o modificados

Esta sección representa una sección de código que realiza pruebas de consistencia sobre campos nuevos o modificados.

NOTA: El programa no modificará un status 'c'

Este bloque representa una sección de código que envía observaciones a NEWMASTR y TX, según lo apropiado

Fig. 15. Procesamiento de EDUPDATE.

3.10 Respaldo y Rotación de Archivos y Denominación de los Archivos Maestros

La Figura anterior ilustra el uso de números en la creación de nombres de archivos para los archivos maestros y de transacción. Nótese que el nombre realmente usado para OLDMASTR es B.MASTER13 y el nombre usado para NEWMASTR es B.MASTER14. Esto indica que OLDMASTR fue el resultado de 13a actualización y la actualización corriente es la 14a, produciendo MASTER14. En este caso, ambos MASTERS están en la misma base de datos, B. (Una mejor estrategia usa dos bases de datos en paquetes de disco separados, especialmente para archivos grandes).

El OLDMASTR se tiene como un sustituto. Si el NEWMASTR es destruido por alguna razón, o si el trabajo de actualización falla, la primera acción a tomarse es copiar (OLDMASTR) (aquí B.MASTER13) a una cinta y colocar esta cinta en un lugar seguro. Esta acción protege al OLDMASTR como un sustituto en caso de ocurrir problemas adicionales. Entonces, después de que el respaldo se complete con éxito, se intenta correr la actualización correctamente.

Si al copiar la cinta se destruye el OLDMASTR (como resultado de un error de sistema, falla eléctrica durante la operación o cualquier otra causa) debe intentarse sacar una copia de respaldo o del OLDMASTR previo (aquí B.MASTER12). Entonces se harán dos corridas de actualización, una para reproducir OLDMASTR y luego otra para producir NEWMASTR.

De vez en cuando se copia el NEWMASTR a una cinta que se coloca en un sitio seguro y remoto. Muchas instituciones hacen esto después de cada ocho actualizaciones (MASTER8, MASTER16, etc.).

Eventualmente se agota el espacio en las bases de datos SAS usadas para los archivos maestros (un PROC CONTENTS debe correrse antes de cada actualización para determinar si hay disponible suficiente espacio). Se logra disponibilidad de espacio eliminando los archivos maestros más antiguos y copiando los archivos de transacciones correspondientes a una cinta. Antes de eliminar algún archivo MAESTRO se copia el MAESTRO maś reciente a una cinta, junto con todos los archivos transacción usados para producirlo. Por ejemplo, supóngase que MASTER8 fue respaldado previamente en cinta y se han corrido actualizaciones para producir MASTER9, MASTER10..., MASTER14. Se necesita ahora, suprimir archivos para hacer espacio disponible para el MASTER15. Primero se copia

a cinta el MASTER14 y todos los archivos de transacción usados para crear MASTER9, MASTER10, ..., MASTER14. Se verifica la legibilidad de esta cinta, copiando su contenido a otra cinta (PROC TAPECOPY). Se verifica la legibilidad de la copia extra y se envía a una localización segura y remota. Ahora MASTER8, MASTER9, ..., MASTER12 (no MASTER13 o MASTER14) y todos los archivos de transacción usados para crear MASTER8, ..., MASTER13 pueden ser eliminados. Siempre se guardan en disco los dos MAESTROS más recientes y los archivos de transacción usados para crearlos.

Este sistema de 'seguro' provee protección contra la destrucción de los archivos MAESTROS, pues sin uso de ellos es destruido accidentalmente, puede volverse a crear con tiempo, esfuerzo y costo razonable.

3.11 El Subsistema de Inventario

3.11.1 Introducción

El Subsistema de Inventario (INVS) es un pequeño sistema de manejo de datos que ayuda, en la gerencia del proyecto de procesamiento de datos, a controlar su inventario de registro de datos. El INVS mantiene información acerca del progreso de cada registro de datos a través de los varios componentes de manipulación de datos. Si el sistema principal de manejo de datos pierde registros o erróneamente 'genera' registros adicionales, esto puede descubirse al examinar los reportes del INVS.

INVS recibe información (observaciones de transacción) de cada una de las siguientes fases del manejo de datos:

1. Observaciones de 'llegada' del Procesamiento Manual Preliminar.
2. Observaciones de 'entrada' del Programa de Fase de Entrada de Datos.
3. Observaciones de 'error' si mensajes de error son generados para la observación de datos, ya sea por la Fase de Entrada de Datos o los programas EDUPDATE.
4. Observaciones de 'corrección' generadas por el programa EDUPDATE cuando transacciones de corrección son actualizadas al archivo maestro.

5. Observaciones de 'actualizar' cuando la observación de datos es agregada al archivo de datos.

Cada transacción INVS contiene la fecha (observaciones de llegada) o fecha y hora (observaciones que la computadora genera) del evento que se está reportando. Así, el progreso de una observación de datos a través del sistema puede ser seguido a través de los datos del INVS (a propósito, las mismas variables se mantienen en los archivos MAESTROS).

Se describe un diagrama de flujo del INVS en la Fig. 16, en la que hay tres fases básicas:

1. Generación, verificación y corrección de registros de llegada en el Procesamiento Manual Preliminar.
2. Generación de otros registros de transacción (por otros programas del subsistema, tales como DIP, EDUPDATE).
3. Mantenimiento de archivos y generación de reportes.

El diagrama de flujo muestra solamente la primera y tercera fases, la segunda es un componente de otros subsistemas.

3.11.2 Generando Observaciones de Llegada

La fase del procesamiento manual preliminar genera información para registros de llegada (INVCARDS en el diagrama de flujo). Los detalles de este componente dependen mucho de los detalles del proyecto dado: la cantidad de formas recibidas y procesadas, el número de fuentes de datos, el personal disponible, otros. El objetivo es obtener información muy exacta acerca de la llegada de cada forma de datos, pero hacerlo con tan poco trabajo y costo como sea posible. Típicamente, cuando las formas llegan desde una localización remota, el centro de procesamiento prepara una 'forma de desempaque' para registrar la llegada de datos; con diseño cuidadoso esta forma puede ser la forma de datos fuente para datos de llegada (INVCARDS). La Fig. 17 ilustra en forma esquemática el aspecto que podría tener tal forma. Nótese que varias páginas de FORMAS DE LLEGADA DE DATOS pueden ser generadas en cualquier día y cada página puede generar varias tarjetas de

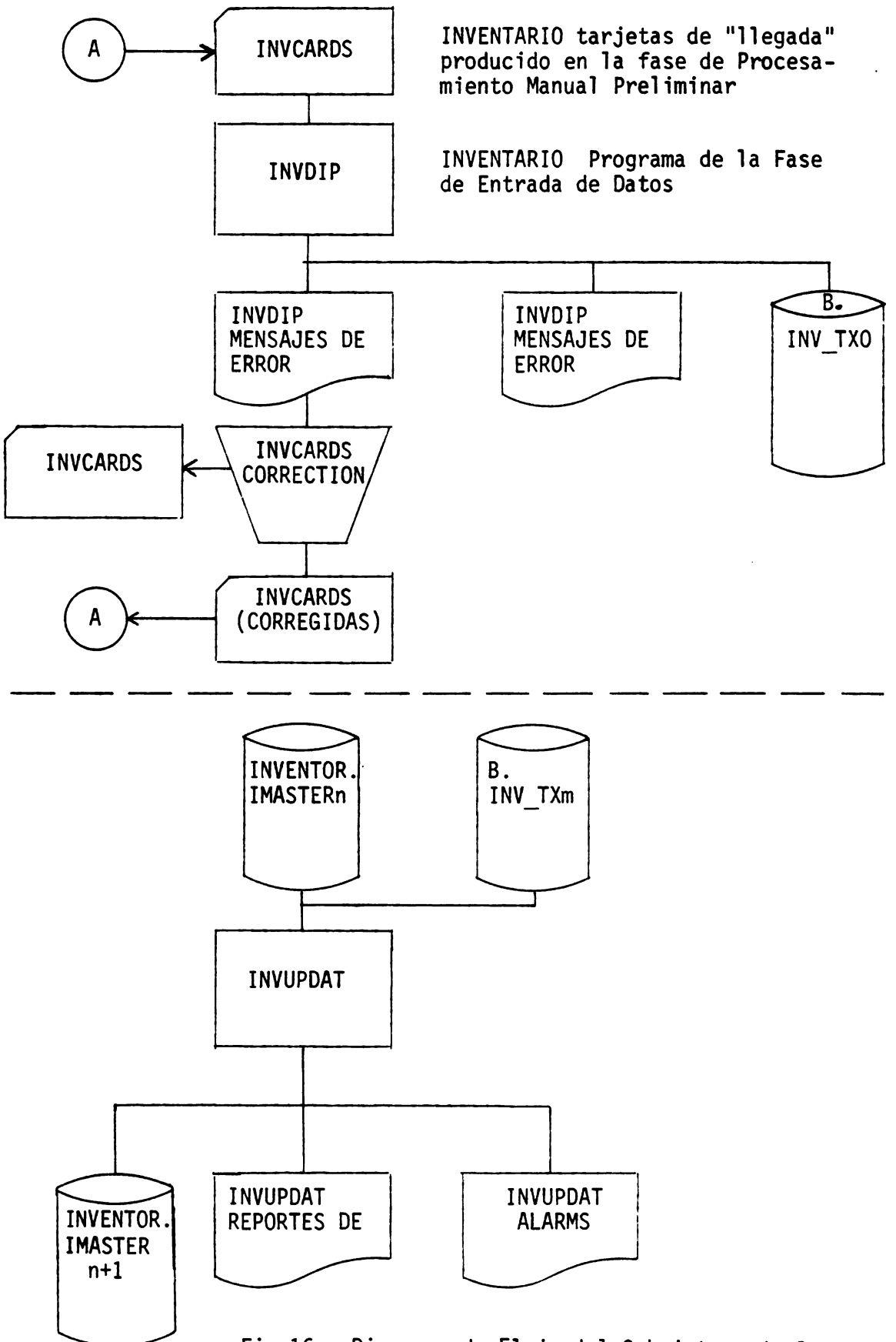


Fig.16. Diagrama de Flujo del Subsistema de Inventario

información. Toda la información de identificación (Proyecto, fecha de llegada, otros) se duplica en cada tarjeta, desde luego.

Las FORMAS DE LLEGADA DE DATOS son revisadas para exactitud e inmediatamente después se perforan (las formas de datos pasan a través de una revisión más extensa). Las tarjetas resultantes, denominadas INVCARDS en el diagrama de flujo, son entonces procesadas por el programa INVDIP de la Fase de Entrada de Datos del INVS.

3.11.3 Procesamiento DIP de INVCARDS

El programa INVDIP lee las INVCARDS y las revisa buscando errores; produce dos archivos de impresión, INVDIP STATUS e INVDIP MENSAJES DE ERROR, y un archivo SAS de salida de transacciones de inventario INVTXO.

INVCARDS. Las INVCARDS son perforadas y revisadas directamente de la forma de llegada de datos; el formato de las INVCARDS obviamente depende del formato de su forma.

El INVDIP STATUS es un archivo de impresión conteniendo dos secciones. La primera es una lista de los datos de entrada en un formato muy semejante a la forma de llegada de datos. La lista es útil para la corrección de datos de entrada en busca de errores. La segunda lista es esencialmente una lista PROC PRINT del archivo de salida, INVTXO; se usa para revisar la salida del programa.

INVDIP MENSAJES DE ERROR. El programa INVDIP realiza una variedad de pruebas de error de variables de datos y de variables clave, pruebas en busca de tarjetas o 'páginas' de entrada extraviadas y otros. Cualesquiera errores o valores improbables son indicados en esta impresión.

El programa INVDIP lee los datos de entrada y produce los impresos ya descritos, el archivo de salida, INVTXO, ordenado por variables clave.

El INVDIP también tiene que ser capaz de procesar los registros de corrección y supresión del INVS, que se usan para corregir errores

que entran en los archivos MAESTROS DEL INVS. Los registros de corrección y supresión del INVS, y registros de 'nueva observación', contiene señales muy parecidas a la variable FLAG en las observaciones de corrección y supresión del DMS.

El INVTXO contiene una observación por cada forma recibida (o error, supresión, o 'nueva' observación con una FLAG). Cada observación tiene:

1. Variable clave (como en los MAESTROS de datos)
2. FORM TYPE - (para proyectos con tipos de formas múltiples)
3. DATEARR - fecha de llegada (variable de fecha SAS)
4. FLAG - para indicar nuevos datos, corrección de error o registro de supresión
5. PAGE - número de página de los datos de entrada
6. CARDNO - número de tarjeta dentro de la página
7. (otras variables en los MAESTROS del INVS)

Nótese que PAGE y CARDNO están disponibles para revisión y no están actualizadas al MAESTRO del INVS.

CORRECCION DE INVCARDS. Después que INDIP ha corrido, la salida es cuidadosamente revisada en busca de mensajes de error y el impreso de datos es comparado contra las formas de llegada de datos. Si se encuentran errores se corrigen las INVCARDS y el trabajo se vuelve a correr. No se mueven datos del INVS a la fase de actualización hasta que todos los errores han sido corregidos.

Este proceso de detección y corrección de error tiene alta prioridad porque los archivos del INVS tienen que ser actualizados rápidamente para que sirva su propósito.

3.11.4 Fase de Mantenimiento de los Archivos del INVS

El mantenimiento de los archivos del INVS es una función SAS de mantenimiento de archivo, basado en la instrucción UPDATE, con una rutina que genera reportes. Como se hizo notar, las instrucciones de actualización del INVS vienen de diferentes fuentes, todas las cuales,

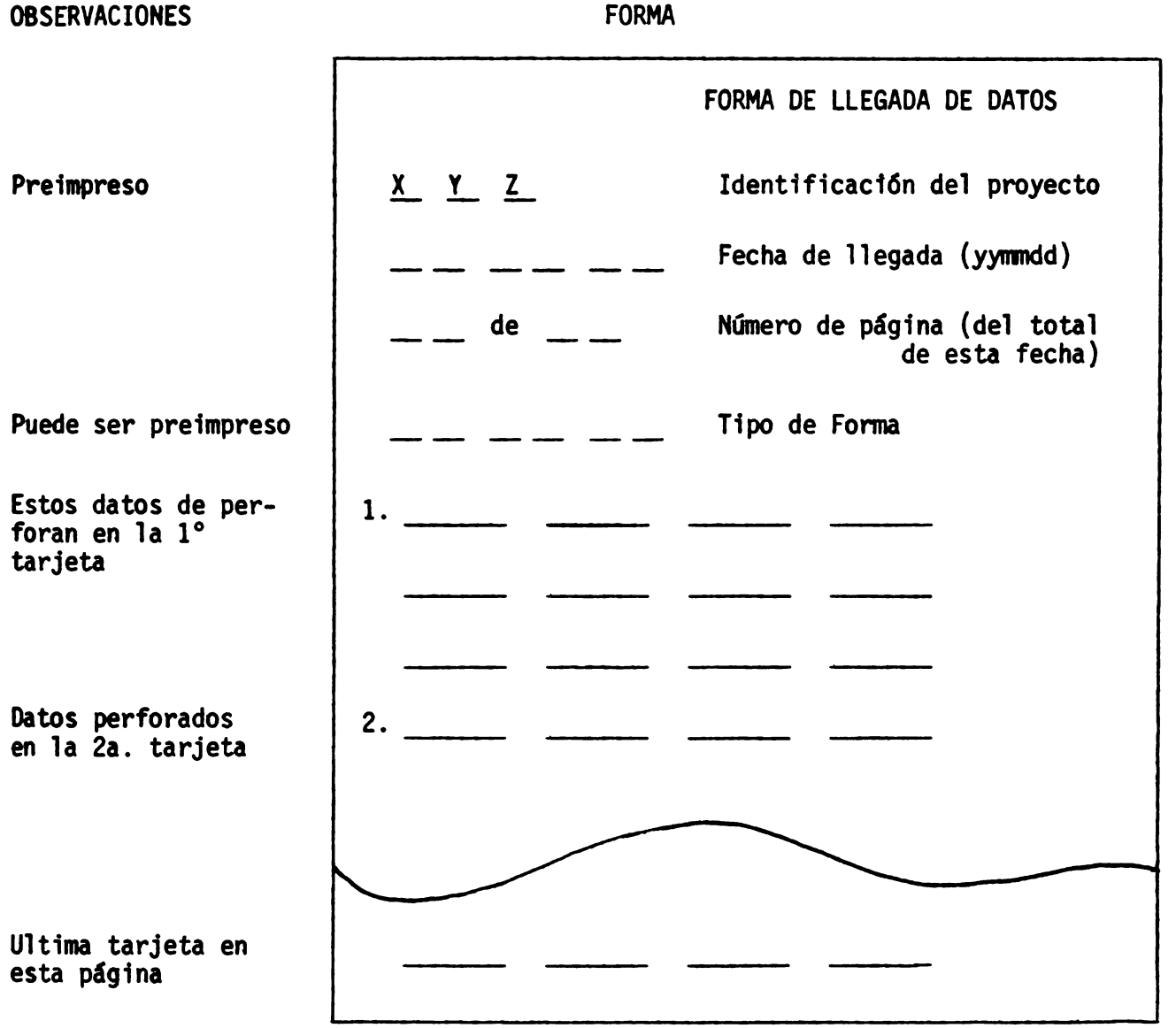


Fig. 17. Ilustración Esquemática de Forma de Llegada Datos

con excepción de una, son generadas por computadora. El programa de actualización, INVUPDAT, no distingue entre fuentes. La mitad inferior de la Fig. 17 tiene una diagrama de flujo general de esta fase. El sistema se describe abajo, en términos de sus entradas, procesamiento y salidas.

INVENTOR.IMASTERn. Los archivos maestros del INVS tienen los mismos campos clave que los archivos maestros de datos, y, en efecto, el último IMASTER contiene:

1. Una observación por cada observación en los datos NEWMASTR, más
2. una observación por cada observación que ha sido suprimida de los archivos maestros de datos.

Los campos de datos IMASTER son:

1. DATEARR -- fecha en que la forma de datos llegó
2. INPUTDAY -- fecha y hora en que la observación de datos fue entrada por el programa DIP.
3. UPDATDAY -- fecha y hora en que la observación fue originalmente agregada al NEWMASTER por el programa EDUPDATE.
4. CHECKDAY -- fecha y hora en que la observación de datos fue revisada más recientemente, en busca de errores de datos por EDUPDAT.
5. CORRDAY -- fecha-hora en que la observación de datos fue corregida más recientemente por EDUPDATE.
6. DELDAY -- fecha-hora en que la observación de datos fue suprimida del archivo maestro por EDUPDATE.
7. ONMASTER -- fecha y hora en que la observación de datos fue observada más recientemente por EDUPDATE, en el archivo MAESTRO de datos. (Esto debe ser la fecha-hora de la ejecución más reciente de EDUPDATE, a menos que la observación haya sido suprimida).
8. CLEAN -- una variable Booleana con valor 1 si todas las variables en la observación MAESTRA de datos tienen status 'b' o 'c'. De lo contrario CLEAN = 0.

Cuando se agrega una nueva observación a IMASTER por un registro de llegada la mayoría de los campos, representando fechas de eventos aun sin realizarse, tendrán valores omitidos. A medida que las transacciones vienen de las varias etapas del procesamiento de los valores omitidos, éstos serán reemplazados con fechas-hora reales.

B.INVTXm. El archivo de transacción del INVS tiene las mismas variables clave que IMASTER y la misma secuencia de ordenamiento. Las otras variables en el archivo de transacción dependen de la fuente de los registros de transacción. Se puede revisar las descripciones de esos componentes para encontrar las descripciones detalladas de los archivos de transacción del INVS.

INVUPDAT es un programa de ACTUALIZACION que también revisa, buscando errores de datos de transacción del INVS, errores de actualización del archivo MAESTRO, y produce reportes de status del procesamiento de datos, los cuales son descritos en los siguientes párrafos. Examinando los reportes de salida se puede determinar fácilmente la estructura del programa.

REPORTES DE STATUS. INVUPDAT produce varios cuadros con el siguiente formato general (Cuadro No. 5).

	Llegada	DIP	UPDATE	CHECK	CORR
Llegada	XXX	XXX	XXX	XXX	XXX
DIP		XXX	XXX	XXX	XXX
UPDATE			XXX	XXX	XXX
CHECK				XXX	XXX
CORR					XXX

Cuadro No. 5. INVUPDAT, reportes de Status

Varios tipos de entradas se hacen con este cuadro. Por ejemplo, una tabulación es el promedio de "tiempos de tránsito", i.e., tiempo para proceder del paso X al paso Y. (e.g., en la primera columna, de llegada a DIP, Llegada a UPDATE, etc.). Otras estadísticas de interés son los tiempos de tránsito máximo, desviación estandard de tiempos de tránsito, etc.

Otro tipo de cuadro podría ser:

Etapa	Número de observaciones		
	Procesadas	Atrasadas	Perdidas
Llegada			
DIP			
UPDATE			
CHECK			
CORR			

Cuadro No. 6. INVUPDAT, reportes de status.

Aquí 'atrasada' significa que a una forma le ha tomado mucho tiempo progresar de un paso al otro. Los criterios reales para el status de 'atrasada' están basados en la experiencia con el sistema.

INVUPDAT ALARMS. Los REPORTES DE STATUS le dicen a la gerencia del proyecto cuántas de las formas tienen problemas; el impreso ALARMS pone en lista la identificación de las formas que, según se determina en esta corrida, tienen problemas (están atrasadas o perdidas). Típicamente, sólo un ALARM es emitido por cada problema que se presenta. Si ALARMS fueran creadas para cada problema en cada actualización, el gerente de datos se varía inundado con ALARMS redundantes y por lo tanto perderían su valor.

Además, la gerencia del proyecto puede especificar la producción de otros informes regularmente por INVUPDAT; informes con propósitos especiales también se preparan sobre una base regular (cada tres meses,

por ejemplo) o sobre una base especial usando IMASTER como un archivo de datos y las capacidades de análisis estadísticos SAS.

3.12 El Subsistema de Corrección de Error

El subsistema de corrección de error es básicamente un subsistema humano. Mensajes de error son generados por EDUPDAT. Los humanos procesan los mensajes, determinan las correcciones, codifican y verifican los datos de corrección. El programa DIP de corrección CORRDIP, es ejecutado para producir un archivo SAS de corrección de transacciones y revisa los datos de corrección, buscando errores. Los resultados impresos por CORRDIP son cuidadosamente examinados para asegurar que las correcciones no introducen otros errores en el archivo de datos. Si se encuentran errores los DATOS DE CORRECCION se corrigen y se vuelve a correr el CORRDIP. Este proceso es repetido hasta que se determina que la transacción de correcciones está enteramente correcta. Solamente entonces es liberado el archivo de transacción de corrección al programa EDUPDAT.

Como se hizo notar anteriormente, el Subsistema de Corrección de Error es un sistema humano; su diseño y ejecución es un problema en ingeniería humana. Los factores que afectan su éxito son los mismos que tienen un efecto mayúsculo en el proyecto total de manejo de datos:

1. Buena ingeniería humana
2. Buena documentación
3. Buen entrenamiento del personal operacional
4. Buen manejo de personal
5. Obtención de personal operaciónal con aptitud para darle la atención apropiada a los detalles
6. Tener recursos adecuados y apropiados disponibles: tiempo, espacio, dinero, gente, equipo, programas y procedimientos. Los problemas de programas y procedimientos no son difíciles --los problemas de la gente son cruciales--, lo cual es cierto para la operación del proceso total del manejo de datos. (Nota: Esta sección está incompleta. El texto adicional se proveerá en una fecha futura).

3.13 El Subsistema de Control de Calidad

(Nota: esta sección todavía no está disponible; será producida en una fecha futura. --RWH, 08 Feb. 80--).

3.14 Seguridad de los Datos

(Nota: Esta sección todavía no está disponible; será producida en una fecha futura. --RWH, 08 Feb. 80).

REFERENCIAS

1. BROOKS, F. P. The mythical man-month. Reading, Mass., Addison-Wesley Pub. Co., 1975.

2. HELMS, R. W. The lipids visit II data management system. Documentation at the Central Patient Registry, Lipids Research Clinic Program. Chapel Hill, University of North Carolina, Dept. of Biostatistics, 1972.

3. _____. An overview of research data management with special emphasis on current problems. Proceedings of the ASA Section on Statistical Computing. Washington, D.C., American Statistical Association, 1978.

4. _____, CHRISTIANSEN, D. H., GALLAGHER, P.N. AND MORRISSEY, L. J. Designing file structures for longitudinal research data. Proceedings of the American Statistical Computing Section. Washington, D.C., American Statistical Association, 1979.

5. METZGER, P. W. Managing a programming project. Englewood Cliffs, N. J., Prentice-Hall, 1973.

6. SAS79. SAS User's Guide. 1979 Ed. Raleigh, N. C., SAS Institute Inc., 1979.

Todas las observaciones en NEWMASTR.

CENTRO INTERAMERICANO DE DOCUMENTACION E INFORMACION AGRICOLA-CIDIA

Serie : Documentación e Información Agrícola

1. Colección de referencia de la Biblioteca Conmemorativa Orton. 2 ed. 1967.
2. Publicaciones periódicas de la Biblioteca Conmemorativa Orton. 1964.
3. Tesis de la Escuela para Graduados 1947-1968; resúmenes. 2 ed. rev. y ampl. 1969.
4. Redacción de referencias bibliográficas; normas oficiales del IICA. 2ed. 1972.
5. Directorio de bibliotecas agrícolas en América Latina. 1964.
6. Catálogo de publicaciones periódicas de la Biblioteca Conmemorativa Orton. 2 ed. rev. y ampl. 1970.
7. Estado actual de bibliotecas agrícolas en América del Sur; resultados de una encuesta personal. 1966.
8. Administración de bibliotecas agrícolas. 1966.
9. Guía de publicaciones periódicas agrícolas de América Latina. 1966.
10. Bibliografía de bibliografías agrícolas de América Latina. 2 ed. rev. y ampl. 1969.
11. I Mesa Redonda sobre el Programa Interamericano de Desarrollo de Bibliotecas Agrícolas, Lima. 1968.
12. Contribuciones del IICA a la literatura de las ciencias agrícolas. 3 ed. rev. 1977.
13. Directorio de siglas en ciencias agrícolas. 2 ed. 1971.
14. Guía básica para bibliotecas agrícolas (ed. en portugués y español). 1969.
15. II Mesa Redonda sobre el Programa Interamericano de Desarrollo de Bibliotecas Agrícolas, Bogotá. 1969.
16. Recursos de bibliotecas agrícolas en América Latina. 1969.
17. 2000 libros en ciencias agrícolas en castellano. 1969.
18. III Mesa Redonda sobre el Programa Interamericano de Desarrollo de Bibliotecas Agrícolas, Río de Janeiro. 1969.
19. Publicaciones periodicas y seriadas de América Latina. 1971.
20. Indice Latinoamericano de tesis agrícolas. 1972.
21. Trópico americano: situación de los servicios bibliotecarios y documentación agrícola. 1972
22. 3000 libros agrícolas en español. 1973.
23. Bibliografía sobre frijol de costa (Vigna sinensis). 1973.
24. Sistema Interamericano de Información para las Ciencias Agrícolas - AGRINTER: bases para su establecimiento. 1973.
25. Bibliografía sobre especies de la fauna silvestre y pesca fluvial y lacustre de América tropical. 1973.
26. Bibliografía sobre plantas de interés económico de la región Amazónica. 1974.

27. Bibliografía sobre sistemas de agricultura tropical. 1974.

28. Bibliografías agrícolas de América Central: PANAMA. Suplemento. 1974.

29. Bibliografía sobre catastro rural en América Latina. 1974.

30. Indice latinoamericano de tesis agrícolas. Suplemento no. 1, 1968-1972. 1974.

31. Bibliografía peruana de pastos y forrajes. 1974.

32. Bibliografías agrícolas de América Central: EL SALVADOR. 1974.

33. Ecología del trópico americano. 1974.

34. Bibliografías agrícolas de América Central: HONDURAS. 1974.

35. Bibliografía selectiva sobre reforma agraria en América Latina 1964-1972. 1974.

36. Manual para Descripción Bibliográfica. Trad. y adapt. del Manual de AGRIS. 2 ed. 1979.

37. Categorías de Materias. Trad. de las Categorías de AGRIS. 2 ed. 1979.

38. Indice de mapas de América Latina y el Caribe existentes en el IICA-CIDIA. 1975.

39. Bibliografías agrícolas de América Central: GUATEMALA. 1975.

40. Bibliografía selectiva sobre derecho y reforma agraria en América Latina, 1972-1974. 1975.

41. La mujer en el medio rural; bibliografía. 1975.

42. Bibliografía colombiana de pastos y forrajes. 1975.

43. Bibliografía sobre silvicultura y ecología forestal tropical. 1975.

44. Silvicultura de bosques tropicales; bibliografía. 1975.

45. Bibliografía internacional sobre la quinua y cañahua. 1976.

46. Bibliografía sobre camélidos sudamericanos. 1976.

47. Bibliografía sobre bovinos criollos de Latinoamérica. 1976.

48. Manual de organización, planificación y operación de los Comités Nacionales de Coordinación (PIADIC). 1976.

49. AGRINTER: origen y evolución; bibliografía anotada. 1976.

50. Bibliografía universitaria de la investigación agrícola en Perú. 1976.

51. Directrices para la selección de documentos en los Sistemas AGRINTER y AGRIS. rev. 1976.

52. Lista de publicaciones periódicas y seriadas. 1976.

53. Bibliografía sobre formas asociativas de producción en el agro. 1977.

54. Camote, maní y soya en América Latina 1970-1975; una bibliografía parcialmente anotada. 1977.

55. Bibliografía sobre aspectos sociales de la producción agropecuaria. 1977.

56. Bibliografía selectiva sobre recursos naturales de Colombia. 1977.

57. Bibliografía colombiana sobre desarrollo rural. 1977.

58. Bibliografía selectiva sobre comercialización agrícola. 1977.

59. Bibliografía sobre reforma agraria en América Latina 1974-1976. 1977.

60. Royas del cafeto (Hemileia spp.) Bibliografía. Suplemento a la 3 ed. 1980.

61. Banco de datos de bibliografías agrícolas de América Latina y el Caribe: Indice acumulado. 1977.

62. Normas de enriquecimiento de títulos. 2 ed. 1980.

63. Vocabulario agrícola en español. 1978.

64. Bibliografía forestal del Perú. 1978.

65. La acción del IICA en el campo de las bibliotecas, documentación e información agrícolas: una síntesis. 1978.

66. Bibliografía sobre ciencias de la información (aportes del IICA). 1978.

67. Bibliografía sobre peste porcina africana. 1979.

68. Centro Interamericano de Documentación, Información y Comunicación Agrícola-CIDIA. 1978.

69. Bibliografía forestal de América tropical. 1979.

70. Bibliografía selectiva sobre desarrollo rural en Venezuela. 1979.

71. Moniliasis: bibliografía. 1979.

72. Bibliografía sobre sensores remotos. 1979.

73. ISIS: manual para usuarios. 1979.

74. Bibliografía básica en desarrollo rural latinoamericano. 1979.

75. Bibliografía sobre desarrollo rural en Ecuador. 1979.

76. Manual para la preparación de perfiles de área para la formulación de alternativas de producción. 1979.

77. Sistema de Información para la Investigación Agropecuaria - SINIA. 1979.

78. Participación de la mujer en el desarrollo rural. 1980.

79. Biomasa y otras fuentes no convencionales de energía: Bibliografía. 1980.

80. Bibliografía sobre colonización en América Latina. 1980.

81. Análisis sobre el desarrollo del Sistema Interamericano de Información Agrícola-AGRINTER. 1980.

82. Rural women: a Caribbean bibliography with special reference to Jamaica. 1980.

83. Bibliografía Agrícola de COSTA RICA. 2 ed. rev. y actualizada. 1980.

84. Documentos producidos por el Fondo Simón Bolívar. 1980.

85. Catálogo colectivo de publicaciones periódicas existentes en bibliotecas agrícolas del Uruguay. 1980.

86. Bibliography of literature relating to research and development in the agricultural sector of Jamaica 1959-1979. 1980.

87. Cáncer de los cítricos. Bibliografía parcialmente anotada. 1980.

88. Rhadinaphelenchus cocophilus "Anillo rojo del cocotero", una bibliografía parcialmente anotada. 1980.

89. Sigatoka del banano: Bibliografía parcialmente anotada. 1980.

90. Mosca del Mediterráneo Ceratitis capitata. Bibliografía parcialmente anotada. 1980.

91. Mulher no Brasil; resumo bibliográfico. 1980.

92. Bibliografía sobre desarrollo rural en Bolivia. 1980.

93. Bibliografía Agrícola del Uruguay 1979-1980. 1981.

94. Páginas de Contenido en Medicina Veterinaria. 1981.

95. Curso corto sobre manejo de datos de investigación usando SAS. Trad. del inglés. 1981.

FECHA DE DEVOLUCION